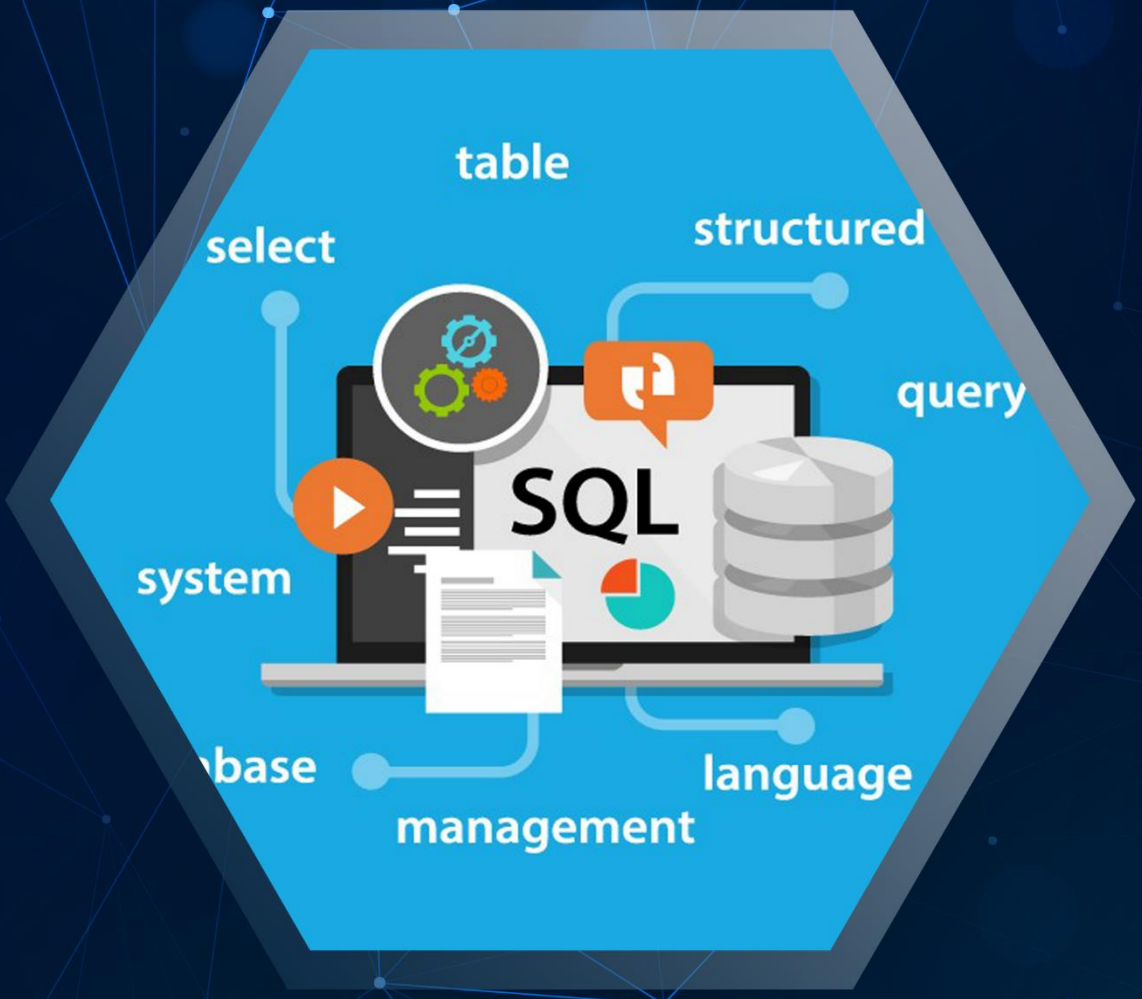
BUKU PT. JNDI

DASAR-DASAR SISTEM BASIS DATA



Yofhanda Septi Eirlangga, S.Kom., M.Kom
Angga Putra Juledi, S.Kom., M.Kom
Kiki Hariani Manurung, S.Kom., M.Kom
Muhammad Thoriq, S,Kom., M.Kom
Aldo Eko Syaputra, S.Kom., M.Kom

Editor : Angga Putra Juledi, S.Kom., M.Kom

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4
Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26
Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

DASAR-DASAR SISTEM BASIS DATA

Yofhanda Septi Eirlangga, S.Kom., M.Kom,
Angga Putra Juledi, S.Kom., M.Kom,
Kiki Hariani Manurung, S.Kom, M.Kom,
Muhammad Thoriq, S.Kom., M.Kom,
Aldo Eko Syaputra, S.Kom., M.Kom

PENERBIT
PT. JASA NIAGA DIGITAL INDONESIA
BUKU PT. JNDI

DASAR-DASAR SISTEM BASIS DATA

Yofhanda Septi Eirlangga, S.Kom., M.Kom,
Angga Putra Juledi, S.Kom., M.Kom,
Kiki Hariani Manurung, S.Kom, M.Kom,
Muhammad Thoriq, S.Kom., M.Kom,
Aldo Eko Syaputra, S.Kom., M.Kom

Desain Cover :
Sahat Parulian Sitorus

Sumber :
https://isbn.jndi.my.id/products/dasar-dasar-sistem-basis-data

Tata Letak :
Nurhasanah

Proofreader :
Sahat Parulian Sitorus

Ukuran :
178, 168, Uk: 21x29 cm

ISBN :
978-623-10-0276-1

Cetakan Pertama :
09 Mei 2024

Hak Cipta 2024, Pada Penulis

Isi diluar tanggung jawab percetakan

Copyright © 2024 by PT.JNDI
All Right Reserved

Hak cipta dilindungi undang-undang
Dilarang keras menerjemahkan, memfotokopi, atau
memperbanyak sebagian atau seluruh isi buku ini
tanpa izin tertulis dari Penerbit.

PENERBIT PT. JASA NIAGA DIGITAL INDONESIA

Jl. H. Maulana Link. VI. Labuhan Ruku, Talawi, Batu Bara, Sumatera Utara 21254
Telp/Wa : 082361274081
https://isbn.jndi.my.id
https://digilib.kampus.jndi.my.id
E-mail: office@jndi.my.id

Kata Pengantar

Selamat datang dalam buku "Dasar-dasar Sistem Basis Data". Buku ini merupakan panduan yang komprehensif untuk memahami konsep, teori, dan praktik terkait dengan sistem basis data dalam konteks dunia teknologi informasi yang terus berkembang pesat.

Dalam era di mana data menjadi salah satu aset paling berharga bagi organisasi dan perusahaan, pemahaman yang kuat tentang sistem basis data menjadi kunci untuk mencapai efisiensi, keamanan, dan inovasi dalam pengelolaan data. Buku ini dirancang untuk memberikan landasan yang kokoh bagi pembaca, baik yang baru mengenal maupun yang sudah berpengalaman dalam bidang ini.

Setiap bab dalam buku ini membahas topik yang relevan dan penting dalam sistem basis data. Dimulai dari pengantar yang membahas peran dan manfaat sistem basis data, hingga bab terakhir yang mengulas tantangan masa depan dalam bidang ini, pembaca akan dibimbing melalui serangkaian materi yang disajikan dengan jelas dan sistematis.

Kami berharap buku ini tidak hanya menjadi sumber pengetahuan yang berharga bagi pembaca, tetapi juga menjadi panduan yang berguna dalam menjelajahi dunia yang kompleks dari sistem basis data. Kami ingin menyampaikan terima kasih kepada semua yang telah berkontribusi dalam pembuatan buku ini, serta kepada pembaca yang telah memberikan kesempatan bagi kami untuk berbagi pengetahuan.

Selamat membaca dan semoga buku ini memberikan manfaat yang besar bagi Anda dalam memahami dan menguasai dasar-dasar sistem basis data.

Rantauprapat, 09 Mei 2024

Tim Penulis

Daftar Isi

Kata Pengantar v
Daftar Isi vi
Daftar Gambar viii
Daftar Tabel x
Bab 1 Pengantar ke Sistem Basis Data 1
1.1 Pengertian Sistem Basis Data 1
1.2 Manfaat Dan Tujuan Basis Data 3
1.3 Sejarah Singkat Sistem Basis Data 4
1.4 Peran dan Manfaat Sistem Basis Data 5
1.5 Kelebihan Dan Kekurangan Basis Data 6
1.6 Tingkatan Data Dalam Database Relasi 7
Bab 2 Model Data dalam Sistem Basis Data 9
2.1 Konsep Model Data 9
2.2 Model relasional 10
2.3 Model hierarki 10
2.4 Model Jaringan 11
2.5 Model database berorientasi objek 12
2.6 Model objek-relasional 12
2.7 Model hubungan entitas 13
2.8 Model basis data lainnya 13
2.9 Model basis data NoSQL 15
Bab 3 Bahasa Pemrograman dalam Sistem Basis Data 16
3.1 Pengantar ke Bahasa Pemrograman Basis Data 16
3.2 SQL (Structured Query Language) 19
3.3 Penggunaan SQL 26
Bab 4 Desain Basis Data 30
4.1 Proses Desain Basis Data 30
4.2 Konsep Normalisasi 47
4.3 Metode Desain Basis Data 52
Bab 5 Implementasi Sistem Basis Data 63
5.1 Tahap Implementas 63

5.2 Pemilihan Perangkat Lunak Basis Data 69
5.3 Instalasi dan Konfigurasi Perangkat Lunak Basis Data 70
5.4 Pengaturan basis data 71
Bab 6 Operasi Sistem Basis Data 74
6.1 Pengelolaan Data 74
6.2 Pencadangan dan Pemulihan Basis Data 75
6.3 Manajemen Transaksi 88
6.4 Pemantauan dan Tuning Kinerja 100
Bab 7 Keamanan Sistem Basis Data 102
7.1 Ancaman Keamanan Data 102
7.2 Strategi Perlindungan Data 109
7.3 Pengaturan Akses Pengguna 114
7.4 Audit dan Pemantauan Keamanan 118
Bab 8 Sistem Basis Data Terdistribusi 122
8.1 Konsep Sistem Terdistribusi 122
8.2 Replikasi Data 129
8.3 Pemulihan Bencana untuk Basis Data 135
8.4 Toleransi kesalahan dalam sistem terdistribusi 143
Bab 9 Big Data dan Sistem Basis Data 147
9.1 Pengantar ke Big Data 147
9.2 Tantangan Penyimpanan dan Analisis Big Data 149
9.3 Solusi Basis Data untuk Big Data 151
9.4 Integrasi Big Data dengan Infrastruktur IT 152
Bab 10 Tantangan Masa Depan dalam Sistem Basis Data 153
10.1 Perkembangan Teknologi Basis Data 153
10.2 Big Data dan Analisis Prediktif 156
10.3 Kecerdasan Buatan dalam Sistem Basis Data 158
10.4 Implikasi Etis dan Hukum 160
Daftar Pustaka 163
Biodata Penulis 166

Daftar Gambar

Gambar 1. 1 Contoh Tingkatan Data Dalam Database Relasi 7
Gambar 2. 1 Contoh Tabel Model relasional 10
Gambar 2. 2 Contoh Gambar Model Hirarki 11
Gambar 2. 3 Contoh Gambar Model Jaringan 11
Gambar 2. 4 Contoh Gambar Model database berorientasi objek 12
Gambar 2. 5 Contoh Model Hubungan Entitas 13
Gambar 3. 1 Contoh Cara Kerja SQL 20
Gambar 3. 2 Contoh Elemen SQL 22
Gambar 3. 3 Contoh Data Cuaca 24
Gambar 3. 4 Contoh Letak Cuaca 25
Gambar 3. 5 Contoh ID Cuaca Tabel Asli 25
Gambar 3. 6 Contoh Penggabungan Kasus 26
Gambar 3. 7 Contoh Join 28
Gambar 4. 1 Contoh Desain Laporan 32
Gambar 4. 2 Contoh Tabel Informasi 34
Gambar 4. 3 Contoh Tabel Database 34
Gambar 4. 4 Contoh Hasil Mengubah Item Informasi Menjadi Kolom 36
Gambar 4. 5 Contoh Tabel Kunci Utama 38
Gambar 4. 6 Contoh Hasil Dari Kunci Utama 38
Gambar 4. 7 Contoh Menghubungkan Tabel 39
Gambar 4. 8 Menghubungkan Satu ke Banyak 40
Gambar 4. 9 Contoh Hasil Dari Satu Menjadi Banyak 40
Gambar 4. 10 Contoh Menghubungkan Banyak ke Banyak 41
Gambar 4. 11 Contoh Hasil Dari Menghubungan Banyak ke Banyak 42
Gambar 4. 12 Contoh Bentuk Tidak Normal 48
Gambar 4. 13 Contoh 1NF / First Normal Form 49
Gambar 4. 14 Contoh 2NF 50
Gambar 4. 15 Contoh 3NF 51
Gambar 4. 16 Contoh Daur Hidup Basis Data 52
Gambar 4. 17 Contoh 6 Tahap Proses Desain Database 53
Gambar 4. 18 Contoh Tipe Entiti Baru 55
Gambar 4. 19 Contoh Tipe Entiti Dalam Dua Tipe Entiti Relasi 55
Gambar 4. 20 Contoh Menemukan dan Menambah Relasi Baru 56
Gambar 4. 21 Contoh Menemukan Katagori Baru Tipe Union 56
Gambar 4. 22 Contoh Fitur Untuk Evaluasi DMBS 58
Gambar 5. 1 Contoh Tabel Admin 63
Gambar 5. 2 Contoh Potongan Script SQL Tabel Admin 63
Gambar 5. 3 Contoh Tabel Barang 64
Gambar 5. 4 Contoh Potongan Script SQL Tabel Barang 64
Gambar 5. 5 Contoh Tabel Detail Pesan 64
Gambar 5. 6 Contoh Potongan Script SQL Tabel Detail Pesan 65

Gambar 5. 7 Contoh Tabel Komentar 65
Gambar 5. 8 Contoh Potomgan Script SQL Tabel Komentar 65
Gambar 5. 9 Contoh Tabel Pemesan 66
Gambar 5. 10 Contoh Potongan Script SQL Tabel Pemesan 66
Gambar 5. 11 Contoh Tabel Member Toko 66
Gambar 5. 12 Contoh Potongan Script SQL Tabel Member Toko 67
Gambar 5. 13 Contoh Tabel User 67
Gambar 5. 14 Contoh Potongan Script SQL Tabel User 67
Gambar 5. 15 Contoh Tabel Suka 68
Gambar 5. 16 Contoh Potongan Script SQL Tabel Suka 68
Gambar 5. 17 Contoh Relasi Tabel Database 68
Gambar 5. 18 Contoh Pengaturan Basis Data 72
Gambar 6. 1 Contoh Diagram State 90
Gambar 6. 2 Contoh Skema shadow-database 92
Gambar 6. 3 Contoh Penjadwalan 1 di text book 93
Gambar 6. 4 Contoh Penjadwalan 3 pada Text book 93
Gambar 6. 5 Contoh Penjadwalan 4 pada text book 94
Gambar 6. 6 Contoh Penjadwalan yang Tidak Conflict Serializable 95
Gambar 6. 7 Contoh conflict serializability 95
Gambar 6. 8 Contoh Penjadwalan yang View Serializable Tapi Tidak Conflict Serializable 96
Gambar 6. 9 Contoh Penjadwalan 9 di Text Book 97
Gambar 6. 10 Contoh Jadwal 11 di Text Book 97
Gambar 6. 11 Contoh Cascading Rollback 98
Gambar 6. 12 Contoh Grafik Preseden Penjadwalan 1 99
Gambar 6. 13 Contoh Grafik Preseden Penjadwalan 4 100
Gambar 7. 1 Contoh Custumer Data 105
Gambar 8. 1 Contoh Basis Data Terdistribusi Homogen 124
Gambar 8. 2 Contoh Sistem Database Terdistribusi Database Oracle 128
Gambar 8. 3 Contoh Replikasi Data 129
Gambar 8. 4 Contoh Replikasi Database NoSQL Write Operation 132
Gambar 8. 5 Contoh Replikasi Database NoSQL Read Operation 132
Gambar 8. 6 Contoh Hasil Replikasi Database NoSQL Write Operation 133
Gambar 8. 7 Contoh HA/DR dan pengukurannya 136
Gambar 8. 8 Contoh Pendekatan pencadangan dan pemulihan DR 137
Gambar 8. 9 Contoh DR aktif-pasif 138
Gambar 8. 10 Contoh Replikasi logis Aurora dengan replika baca lintas wilayah 138
Gambar 8. 11 Contoh Arsitektur penyimpanan TiDB 140
Gambar 8. 12 Contoh Solusi pemulihan bencana multi-wilayah TiDB 140
Gambar 8. 13 Contoh DR multi-wilayah TiDB dengan TiCDC 142
Gambar 9. 1 Contoh Big Data 150

Daftar Tabel

Tabel 5. 1 Item dan Keterangan 73

Bab 1 Pengantar ke Sistem Basis Data

1.1 Pengertian Sistem Basis Data

Berbicara tentang basis data dapat diartikan bahwa seluruh data yang disimpan dalam sebuah basis data ditempatkan pada masing-masing table/file sesuai dengan fungsinya. Dengan tersimpannya data dalam basis data tersebut, maka akan dengan mudah dapat melakukan penelusuran data yang diinginkan sehingga berdampak pada waktu pencarian yang lebih efisien.

Di dalam suatu media penyimpanan (misalnya harddisk), dapat ditempatkan lebih dari 1(satu) basis data secara elektronik. Namun, tidak semua bentuk penyimpanan data yang disimpan secara elektronik dikatakan basis data karena ketika menyimpan dokumen di dalam sebuah harddisk, harddisk tersebut dapat berisi data file teks dari program pengolahan kata, spreadsheet, dan lainnya.

Yang ditekankan dalam basis data adalah pengaturan, pemilahan, pengelompokan, dan pengorganisasian data yang disimpan sesuai dengan fungsinya. Hal tersebut bisa berbentuk sejumlah file, table terpisah, atau dalam bentuk pendefinisian kolom (field) data dalam setiap file atau table tersebut.

Pengertian Basis Data

Basis Data berasal dari kata Basis dan Data. Adapun pengertian dari kedua kata tersebut adalah sebagai berikut:

- Basis dapat diartikan sebagai markas atau gudang atau tempat bersarang atau tempat berkumpul.
- Data dapat diartikan sebagai representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, dan pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya. Objek tersebut direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Basis Data sebagai satu kesatuan dapat didefinisikan sebagai berikut:

- Himpunan kelompok data yang saling berhubungan dan terorganisasi dengan baik agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
- Kumpulan data yang saling berhubungan dan disimpan dengan baik secara bersama-sama tanpa pengulangan (redudansi) yang tidak diperlukan.
- Kumpulan file, tabel, atau arsip yang saling berhubungan dan disimpan dalam satu media penyimpanan elektronik. Kumpulan file ini selanjutnya disebut Tabel (Table) sebagai komponen utama untuk membangun basis data.

Basis Data dapat dipahami sebagai suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media. Walaupun disimpan secara bersamasama dan saling terhubung, kumpulan data tersebut tersimpan tanpa saling tumpang tindih satu sama lain atau tidak terjadi kerangkapan data. Namun, jika pun terjadi kerangkapan data maka kerangkapan data tersebut harus terjadi seminimal mungkin dan dapat terkontrol. Beberapa kondisi data di dalam suatu basis data diantaranya:

- Data disimpan dengan cara-cara tertentu sehingga memudahkan ketika akan digunakan atau ditampilkan kembali.
- Data dapat digunakan oleh satu atau beberapa program aplikasi secara optimal.
- Data disimpan tanpa mengalami ketergantungan dengan program-program yang akan menggunakannya.
- Data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

Dari definisi tersebut dapat disimpulkan bahwa pengertian basis data adalah koleksi terpadu dari data yang saling berkaitan dan dirancang untuk memenuhi kebutuhan informasi suatu organisasi. Masing–masing table/file di dalam basis data tersebut berfungsi untuk menampung atau menyimpan data dimana data-data tersebut saling berhubungan dengan satu dengan yang lain. Dari pengertian tersebut dapat dikatakan bahwa basis data memiliki beberapa kriteria penting antara lain:

- Beorientasi pada data.
- Data dapat digunakan oleh pemakai yang berbeda-beda atau beberapa program aplikasi tanpa perlu mengubah basis data.
- Data dalam basis data dapat berkembang dengan mudah baik volume maupun strukturnya.
- Data yang ada dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
- Data dapat digunakan dengan cara yang berbeda-beda.
- Kerangkapan data minimal.

Sistem Basis Data

Sistem Basis Data atau dikenal dengan Database System didalam Bahasa Ingris. Secara harfiah, sistem basis data terdiri dari dua kata, sistem dan basis data. Oxford memberikan Definisi dari sistem yaitu “an organized set of ideas or theories or a particular way of doing something" atau “kumpulan ide atau teori atau cara tertentu dalam melakukan sesuatu”.

Sedangkan definisi dari basis data, oxford menjelaskan “database is an organized set of data that is stored in a computer and can be looked at and used in various ways” atau “kumpulan data terorganisir yang disimpan di komputer dan dapat dilihat serta digunakan dengan berbagai cara”.

Dalam Kamus Besar Bahasa Indonesia (KBBI), Sistem Adalah “perangkat unsur yang secara teratur saling berkaitan sehingga membentuk suatu totalitas”.

Pengertian sistem basis data secara umum adalah sistem yang terdiri atas kumpulan tabel data yang saling berhubungan dalam sebuah basis data di sebuah sistem komputer dan sekumpulan program yang biasa disebut DBMS/Database Management System yang memungkinkan beberapa pemakai dan/atau program lain untuk mengakses dan memanipulasi tabel-tabel data.

Dapat diambil kesimpulan bahwa, pengertian sistem basis data adalah komponen atau mekanisme proses yang berjalan secara bersamaan dalam pengelolaan kumpulan fakta nyata yang terorganisir saling berelasi di dalam perangkat komputer.

DBMS merujuk kepada alat bantu atau perangkat lunak pengelola basis data, Tentu saja DBMS merupakan termasuk dalam sistem basis data. DBMS membantu Sistem Basis data dalam mengelola basis data.

Berbeda dengan rujukan luar, pemahaman mereka istilah Sistem Basis Data merujuk kepada Database Management System (DBMS), termasuk juga pengertian, definisi, komponen dan fungsi dari sistem basis data.

1.2 Manfaat Dan Tujuan Basis Data

Manfaat basis data adalah untuk pengelolaan data dalam memudahkan atau menemukan kembali data yang dicari dengan cepat. Tujuan Basis Data antara lain sebagai berikut:

1. Kecepatan dan kemudahan (Speed). Dengan basis data dapat menyimpan data atau melakukan perubahan, penghapusan, penambahan, dan pemanggilan kembali data yang tersimpam dengan cepat dan mudah.

2. Efisiensi ruang penyimpanan (Space). Dengan basis data penggunaan ruang penyimpanan data dapat dilakukan dengan melakukan meminimalisasi jumlah pengulangan data dan dengan menerapkan sejumlah pengkodean.

3. Keakuratan (Accuracy). Dengan memanfaatkan pengkodean atau pembentukan relasi antar data, penerapan aturan atau batasan tipe data dapat diterapkan dalam basis data yang berguna untuk menentukan keakuratan saat input data atau penyimpanan data.

4. Keamanan (Security). Sejumlah sistem (aplikasi) pengelolaan basis data tidak menerapkan aspek keamanan dalam penggunaannya. Akan tetapi, untuk sistem yang besar dan serius, aspek keamanan menjadi hal yang penting untuk diterapkan. Dengan begitu, sistem dapat menentukan siapa yang boleh menggunakan basis data dan menentukan jenis operasi-operasi apa saja yang boleh dilakukan.

5. Terpeliharanya keselarasan data (Consistent). Apabila ada perubahan data pada aplikasi yang berbeda, secara otomatis perubahan itu berlaku untuk keseluruhan.

6. Kebersamaan pemakaian (Sharebility). Data dapat dipakai secara bersama-sama oleh beberapa program aplikasi saat bersamaan.

7. Dapat diterapkan standarisasi (Standardization). Dengan adanya pengontrolan yang terpusat, basis data dapat menerapkan standarisasi data yang disimpan sehingga memudahkan pemakaian, distribusi, maupun pertukaran data.

8. Ketersediaan (Availability). Basis data dapat memilah data utama atau master, transaksi, data history hingga data kedaluwarsa. Data yang jarang atau tidak digunakan lagi dapat diatur untuk dipisahkan dari sistem basis data yang aktif.

9. Kelengkapan (Completeness). Kelengkapan sebuah data bersifat relatif, dalam sebuah basis data penilaian kelengkapan data sangat bergantung pada pengguna sehingga penilaian tidak selalu sama.

1.3 Sejarah Singkat Sistem Basis Data

- Pra-Sejarah (Pra-1960an):

Konsep penyimpanan data telah ada sejak manusia mulai mencatat informasi secara tertulis. Sebelum komputer, data disimpan dalam bentuk fisik seperti buku catatan, gulungan papirus, atau bahkan dinding gua.

- Era Awal Komputer (1960-an):

Konsep awal basis data muncul bersamaan dengan pengembangan komputer. Pada tahun 1960-an, Edgar F. Codd mulai mengembangkan model relasional yang menjadi dasar bagi sistem basis data relasional modern.

- 1970-an: Era Basis Data Relasional:

Pada tahun 1970, Edgar F. Codd mempublikasikan makalah yang merinci model relasional pertamanya. IBM memperkenalkan sistem basis data relasional pertama, System R, pada pertengahan tahun 1970-an.

- 1980-an: Pertumbuhan Perangkat Lunak Basis Data:

Pada dekade ini, popularitas sistem basis data relasional mulai meningkat. Oracle Corporation dan Microsoft memperkenalkan produk-produk basis data relasional mereka, Oracle dan SQL Server, yang menjadi pionir dalam industri.

- 1990-an: Era Basis Data Terdistribusi:

Dengan perkembangan internet dan kebutuhan akan akses data yang terdistribusi, sistem basis data terdistribusi mulai berkembang. Teknologi seperti database terdistribusi dan replikasi data menjadi semakin penting.

- 2000-an: Era Big Data:

Pada awal abad ke-21, data mulai tumbuh secara eksponensial, dengan munculnya konsep Big Data. Teknologi seperti Hadoop dan NoSQL databases (Not Only SQL) berkembang untuk mengelola volume data yang besar dengan skala yang lebih luas.

- 2010-an: Konvergensi Big Data dan Analitik:

Penekanan pada analisis data untuk mendapatkan wawasan yang berharga mulai mendominasi. Perusahaan mengintegrasikan sistem basis data tradisional dengan alat analitik dan kecerdasan buatan (AI) untuk mengoptimalkan pengambilan keputusan.

- 2020-an: Arah Baru Teknologi Basis Data:

Pada dekade ini, perkembangan dalam bidang seperti database berbasis cloud, database grafik, dan blockchain menjadi perhatian utama. Kecerdasan buatan semakin diintegrasikan dalam sistem basis data untuk meningkatkan efisiensi dan keamanan.

1.4 Peran dan Manfaat Sistem Basis Data

Sistem Basis Data memiliki peran yang sangat penting dalam dunia teknologi informasi dan bisnis modern. Salah satu peran utamanya adalah sebagai penyimpan dan pengelola data secara terstruktur dan terorganisir.

Dengan menggunakan sistem basis data, organisasi dapat menyimpan berbagai jenis data, mulai dari informasi pelanggan dan produk hingga transaksi keuangan dan inventaris. Ini memungkinkan akses yang cepat dan efisien terhadap informasi yang diperlukan, baik oleh pengguna internal maupun eksternal, seperti manajer, karyawan, atau pelanggan.

Selain itu, sistem basis data juga memungkinkan integrasi data yang lebih baik di seluruh organisasi, memastikan konsistensi dan akurasi informasi yang digunakan untuk pengambilan keputusan. Dengan analisis data yang tepat, organisasi dapat mendapatkan wawasan berharga tentang tren pasar, perilaku pelanggan, dan kinerja operasional mereka. Hal ini memungkinkan mereka untuk mengambil keputusan yang lebih cerdas dan strategis.

Selain itu, sistem basis data juga memainkan peran penting dalam menjaga keamanan dan integritas data. Dengan mengimplementasikan kontrol akses yang tepat dan menjaga kebenaran data, organisasi dapat melindungi informasi sensitif mereka dari akses yang tidak sah atau perubahan yang tidak diinginkan. Secara keseluruhan, sistem basis data tidak hanya berperan sebagai penyimpan data, tetapi juga sebagai alat yang kuat untuk mendukung operasi sehari-hari, pengambilan keputusan, dan inovasi di berbagai sektor industri.

1.5 Kelebihan Dan Kekurangan Basis Data

Kelebihan:

1. Dapat meningkatkan kemandirian data. Sebuah basis data dapat digunakan untuk bermacam-macam program aplikasi tanpa harus mengubah format data yang sudah ada.

2. Konsistensi data. Konsistensi data di dalam basis data dilakukan dengan cara data disimpan hanya sekali dalam basis data sehingga jika terjadi perubahan pada nilai data tersebut, perubahan hanya dilakukan satu kali dan nilai baru tersebut akan tersedia untuk semua pengguna.

3. Meningkatkan aksesibilitas terhadap data dan respon yang lebih baik. Dengan basis data maka aksesibilitas data dan respon akan lebih baik. Hal tersebut dapat dicapai dengan integrasi data yang melewati batasan-batasan departemen dalam organisasi sehingga data dapat langsung diakses oleh pengguna.

4. Pengendalian terhadap kerangkapan data. Data dalam sebuah basis data dilakukan penyimpanan dengan cara disimpan satu kali. Hal ini mengurangi kerangkapan data dan mengurangi biaya untuk tempat penyimpanan.

5. Meningkatkan keamanan data. Keamanan basis data dapat melindungi basis data dari pengguna yang tidak memiliki otorisasi. Basis data dapat menentukan batasan-batasan pengaksesan data, misalnya dengan memberikan password dan pemberian hak akses bagi pemakai (misalnya untuk hak akses dalam proses update, delete, insert, maupun select).

6. Memperbaiki integritas data. Intergritas data mengacu pada validitas dan konsistensi dari data yang disimpan. Integritas biasanya diekspresikan dalam batasan (constraints) yang merupakan aturan yang konsisten dan tidak dapat dilanggar. Jika kerangkapan data dapat dikontrol dan kekonsistenan data dapat dijaga, maka data menjadi akurat.

7. Data dapat dipakai secara bersama-sama. Data yang ada pada basis data menjadi milik seluruh organisasi dan dapat dipakai secara bersama oleh pengguna yang berwenang pada saat bersamaan.

8. Memperoleh lebih banyak informasi dari data yang sama. Pengguna basis data dapat memperoleh informasi selain dari informasi rutin yang dikelolanya karena semua data lain berada dalam basis data yang sama. Dengan demikian, kebutuhan akan informasi selain dari informasi rutin dapat terpenuhi.

Kekurangan:

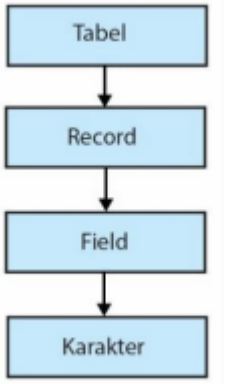
1. Biayanya dapat menjadi sangat mahal karena menyangkut biaya-biaya untuk pembelian sekaligus perawatan hardware dan software. Selain itu, terdapat juga biaya tambahan untuk untuk penyimpanan (storage), jaringan (network), dan lain-lain.

2. Rumit. Perancang, pengembang, Data Base Administator (DBA), dan pengguna akhir harus memahami secara rinci dan mendalam tentang fungsi basis data yang ditangani agar dapat mengambil manfaat dari basis data. Kegagalan dalam memanfaatkannya dapat menyebabkan kerugian yang cukup besar bagi organisasi atau perusahaan.

3. Tambahan biaya konversi. Diperlukan biaya yang besar untuk berpindah dari aplikasi atau sistem yang lama ke dalam sistem basis data yang baru. Selain itu, diperlukan pula biaya untuk pelatihan staf dalam menggunakan sistem yang baru ini serta tambahan biaya untuk mempekerjakan staf khusus seperti DBA, dan lain-lain.

1.6 Tingkatan Data Dalam Database Relasi

Dalam suatu sistem database relasi, data yang tersimpan dalam DBMS mempunyai tingkatan-tingkatannya, sebagaimana tampak dalam Gambar 1.1 berikut:



Gambar 1. 1 Contoh Tingkatan Data Dalam Database Relasi

a. Karakter (character) Merupakan bagian terkecil dalam database, dapat berupa karakter numerik (angka 0 s.d 9), huruf (A – Z, a – z) ataupun karakter-karakter khusus, seperti *, &. %, # dan lain-lain.

b. Field atau Attribute Merupakan bagian dari record yang menunjukkan suatu item data yang sejenis, misalnya field nama, field NIM, dan lain sebagainya. Setiap field harus mempunyai nama dan tipe data tertentu. Isi dari field disebut Data Value. Dalam table dari sebuah database, field ini disebut juga kolom.

c. Record atau Tupple Tuple/Record adalah kumpulan data value dari attribute yang berkaitan sehingga dapat menjelaskan sebuah entity secara lengkap. Misal record entity mahasiswa adalah kumpulan data value dari field nomor telepon genggam, nama, jurusan, dan alamat per-barisnya. Dalam database, record disebut juga baris.

d. Table/Entity Entity merupakan sesuatu yang dapat diidentifikasi dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian, atau konsep yang informasinya akan disimpan di database. Misalnya pada sistem database akademik, yang menjadi entity adalah mahasiswa, dosen, mata kuliah, dan lain-lain. Dalam aplikasinya, penggunaan istilah entity sering disamakan dengan istilah tabel (entity = table). Disebut table karena dalam merepresentasikan datanya diatur dalam bentuk baris dan kolom. Baris mewakili 1 record dan kolom mewakili 1 field. Kemudian dalam sistem database tradisional, entity atau table ini disebut juga dengan file. Ada beberapa sifat yang melekat pada suatu tabel yaitu:

    - Tidak boleh ada record yang sama atau kembar.
    - Urutan record tidak terlalu penting karena data dalam record dapat diurutkan sesuai dengan kebutuhan.
    - Setiap field harus mepunyai nama yang unik atau tidak boleh ada yang sama.
    - Setiap field mesti mempunyai tipe data dan karakteristik tertentu.

Bab 2 Model Data dalam Sistem Basis Data

2.1 Konsep Model Data

Model database menunjukkan struktur logis dari database, termasuk hubungan dan batasan yang menentukan bagaimana data dapat disimpan dan diakses. Model database individual dirancang berdasarkan aturan dan konsep model data mana pun yang lebih luas yang diadopsi oleh perancang. Kebanyakan model data dapat direpresentasikan dengan diagram database yang menyertainya.

Jenis model basis data

Ada banyak jenis model data. Beberapa yang paling umum meliputi:

- Model basis data hierarki
- Model relasional
- Model jaringan
- Model database berorientasi objek
- Model hubungan entitas
- Model dokumen
- Model nilai atribut-entitas
- Skema bintang
- Model objek-relasional, yang menggabungkan dua hal yang membentuk namanya

Anda dapat memilih untuk menggambarkan database dengan salah satu dari ini tergantung pada beberapa faktor. Faktor terbesarnya adalah apakah sistem manajemen database yang Anda gunakan mendukung model tertentu. Sebagian besar sistem manajemen basis data dibangun dengan mempertimbangkan model data tertentu dan mengharuskan penggunanya untuk mengadopsi model tersebut, meskipun beberapa sistem mendukung banyak model.

Selain itu, model yang berbeda berlaku untuk tahapan proses desain database yang berbeda. Model data konseptual tingkat tinggi paling baik untuk memetakan hubungan antar data dengan cara orang memandang data tersebut. Model logis berbasis catatan, di sisi lain, lebih mencerminkan cara data disimpan di server.

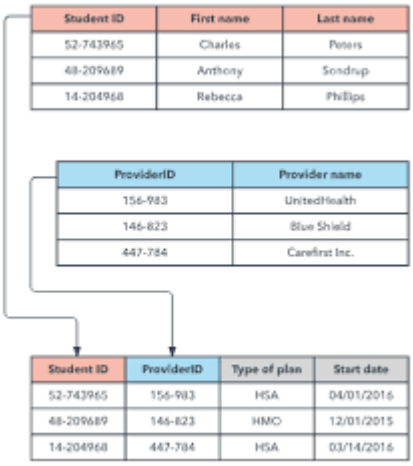
Memilih model data juga merupakan masalah menyelaraskan prioritas database Anda dengan kekuatan model tertentu, apakah prioritas tersebut mencakup kecepatan, pengurangan biaya, kegunaan, atau hal lainnya.

2.2 Model relasional

Model yang paling umum, model relasional mengurutkan data ke dalam tabel, disebut juga relasi, yang masing-masing terdiri dari kolom dan baris. Setiap kolom mencantumkan atribut entitas yang dimaksud, seperti harga, kode pos, atau tanggal lahir. Bersama-sama, atribut-atribut dalam suatu relasi disebut domain. Atribut atau kombinasi atribut tertentu dipilih sebagai kunci utama yang dapat dirujuk dalam tabel lain, bila disebut kunci asing.

Setiap baris, juga disebut tupel, mencakup data tentang contoh spesifik dari entitas yang bersangkutan, seperti karyawan tertentu.

Model ini juga memperhitungkan tipe hubungan antara tabel-tabel tersebut, termasuk hubungan satu-ke-satu, satu-ke-banyak, dan banyak-ke-banyak. Berikut ini contohnya:



Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

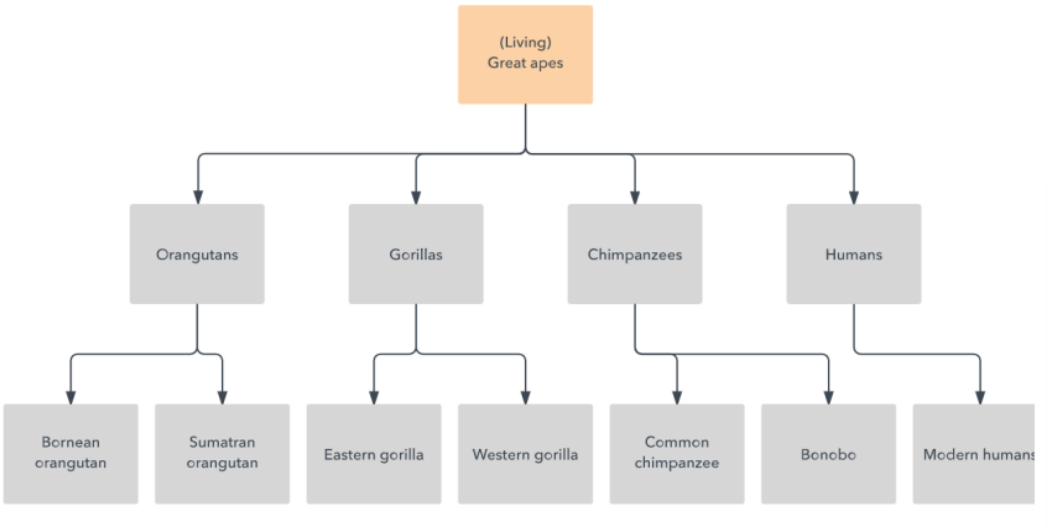
Gambar 2. 1 Contoh Tabel Model relasional

Di dalam database, tabel dapat dinormalisasi, atau disesuaikan dengan aturan normalisasi yang membuat database fleksibel, mudah beradaptasi, dan terukur. Saat dinormalisasi, setiap bagian data bersifat atomik, atau dipecah menjadi bagian terkecil yang berguna.

Basis data relasional biasanya ditulis dalam Structured Query Language (SQL). Model ini diperkenalkan oleh EF Codd pada tahun 1970.

2.3 Model hierarki

Model hierarki mengatur data ke dalam struktur seperti pohon, di mana setiap record memiliki satu induk atau akar. Catatan saudara diurutkan dalam urutan tertentu. Perintah itu digunakan sebagai perintah fisik untuk menyimpan database. Model ini bagus untuk menggambarkan banyak hubungan di dunia nyata.



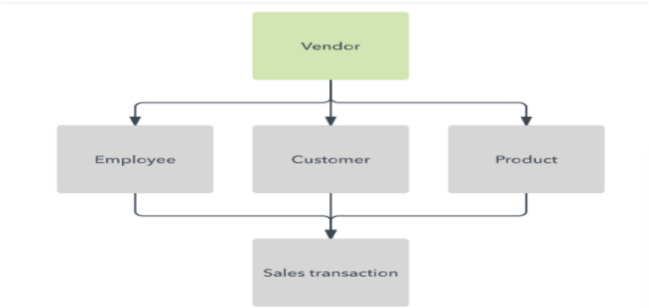
Gambar 2. 2 Contoh Gambar Model Hirarki

Model ini terutama digunakan oleh Sistem Manajemen Informasi IBM pada tahun 60an dan 70an, namun model ini jarang terlihat saat ini karena inefisiensi operasional tertentu.

2.4 Model Jaringan

Model jaringan dibangun berdasarkan model hierarki dengan memungkinkan hubungan banyak-ke-banyak antara catatan-catatan tertaut, yang menyiratkan beberapa catatan induk. Berdasarkan teori himpunan matematika, model dibangun dengan kumpulan catatan terkait. Setiap set terdiri dari satu catatan pemilik atau induk dan satu atau lebih catatan anggota atau anak. Sebuah record dapat menjadi anggota atau anak dalam beberapa set, sehingga model ini dapat menyampaikan hubungan yang kompleks.

Bahasa ini paling populer pada tahun 70an setelah ditetapkan secara resmi oleh Conference on Data Systems Languages (CODASYL).



Gambar 2. 3 Contoh Gambar Model Jaringan

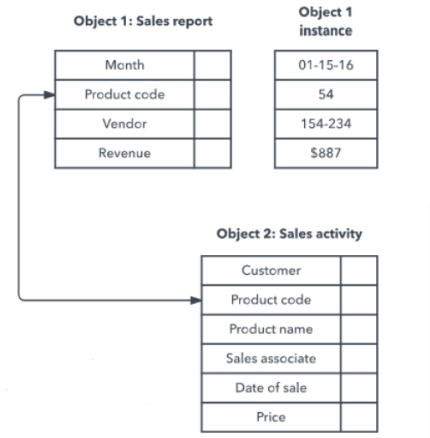
2.5 Model database berorientasi objek

Model ini mendefinisikan database sebagai kumpulan objek, atau elemen perangkat lunak yang dapat digunakan kembali, dengan fitur dan metode terkait. Ada beberapa jenis database berorientasi objek:

Basis data multimedia menggabungkan media, seperti gambar, yang tidak dapat disimpan dalam database relasional.

Basis data hiperteks memungkinkan objek apa pun untuk terhubung ke objek lain. Ini berguna untuk mengatur banyak data yang berbeda, namun tidak ideal untuk analisis numerik.

Model database berorientasi objek adalah model database pasca-relasional yang paling terkenal, karena model ini menggabungkan tabel, namun tidak terbatas pada tabel. Model seperti ini juga dikenal sebagai model database hybrid.



Gambar 2. 4 Contoh Gambar Model database berorientasi objek

2.6 Model objek-relasional

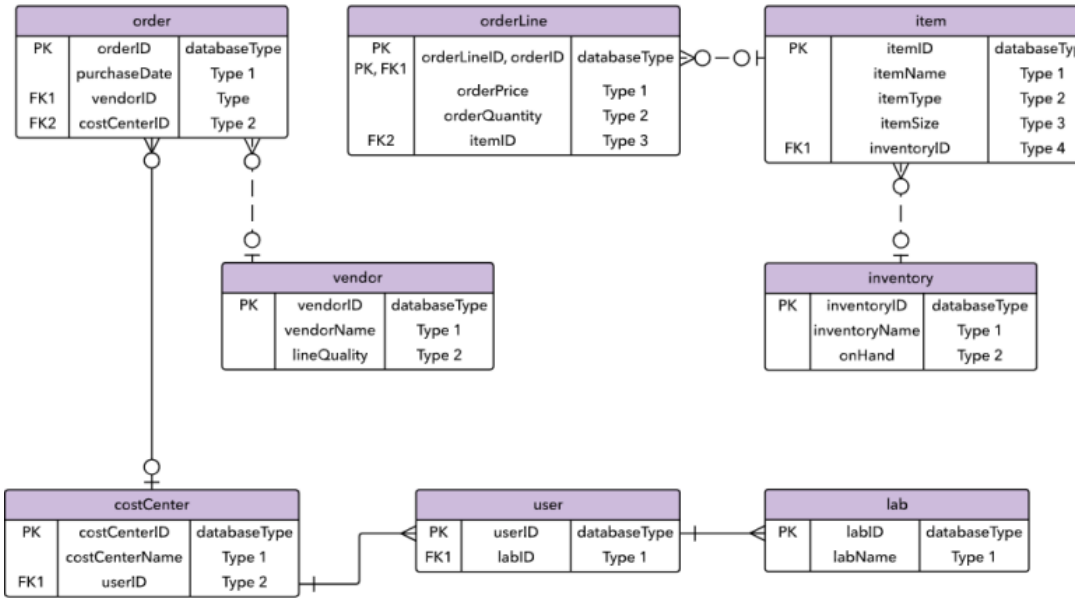
Model database hybrid ini menggabungkan kesederhanaan model relasional dengan beberapa fungsionalitas tingkat lanjut dari model database berorientasi objek. Intinya, ini memungkinkan desainer untuk memasukkan objek ke dalam struktur tabel yang sudah dikenalnya.

Bahasa dan antarmuka panggilan mencakup SQL3, bahasa vendor, ODBC, JDBC, dan antarmuka panggilan berpemilik yang merupakan ekstensi dari bahasa dan antarmuka yang digunakan oleh model relasional.

2.7 Model hubungan entitas

Model ini menangkap hubungan antara entitas dunia nyata seperti model jaringan, namun tidak terkait langsung dengan struktur fisik database. Sebaliknya, ini sering digunakan untuk mendesain database secara konseptual.

Di sini, orang, tempat, dan benda tempat titik data disimpan disebut sebagai entitas, yang masing-masing memiliki atribut tertentu yang bersama-sama membentuk domainnya. Kardinalitas, atau hubungan antar entitas, juga dipetakan.



Gambar 2. 5 Contoh Model Hubungan Entitas

Bentuk umum dari diagram ER adalah skema bintang, di mana tabel fakta pusat terhubung ke tabel beberapa dimensi.

2.8 Model basis data lainnya

Berbagai model database lain telah atau masih digunakan sampai sekarang.

- **Model file terbalik**

Basis data yang dibangun dengan struktur file terbalik dirancang untuk memfasilitasi pencarian teks lengkap dengan cepat. Dalam model ini, konten data diindeks sebagai serangkaian

kunci dalam tabel pencarian, dengan nilai yang menunjuk ke lokasi file terkait. Struktur ini dapat memberikan pelaporan yang hampir seketika dalam big data dan analitik, misalnya.

Model ini telah digunakan oleh sistem manajemen database ADABAS Software AG sejak tahun 1970, dan masih didukung hingga saat ini.

- **Model datar**

Model datar adalah model data yang paling awal dan paling sederhana. Ini hanya mencantumkan semua data dalam satu tabel, yang terdiri dari kolom dan baris. Untuk mengakses atau memanipulasi data, komputer harus membaca seluruh file datar ke dalam memori, sehingga model ini tidak efisien untuk semua kecuali kumpulan data terkecil.

- **Model multidimensi**

Ini adalah variasi dari model relasional yang dirancang untuk memfasilitasi pemrosesan analitis yang lebih baik. Meskipun model relasional dioptimalkan untuk pemrosesan transaksi online (OLTP), model ini dirancang untuk pemrosesan analitis online (OLAP).

Setiap sel dalam database dimensi berisi data tentang dimensi yang dilacak oleh database. Secara visual, ini seperti kumpulan kubus, bukan tabel dua dimensi.

- **Model semi terstruktur**

Dalam model ini, data struktural yang biasanya terdapat dalam skema database disematkan dengan data itu sendiri. Di sini perbedaan antara data dan skema tidak jelas. Model ini berguna untuk mendeskripsikan sistem, seperti sumber data berbasis web tertentu, yang kami perlakukan sebagai database tetapi tidak dapat dibatasi dengan skema. Ini juga berguna untuk menjelaskan interaksi antar database yang tidak mengikuti skema yang sama.

- **Model konteks**

Model ini dapat menggabungkan elemen dari model database lain sesuai kebutuhan. Ini menyatukan elemen-elemen dari model berorientasi objek, semi terstruktur, dan jaringan.

- **Model asosiatif**

Model ini membagi semua titik data berdasarkan apakah titik tersebut menggambarkan suatu entitas atau asosiasi. Dalam model ini, entitas adalah segala sesuatu yang ada secara mandiri, sedangkan asosiasi adalah sesuatu yang hanya ada dalam kaitannya dengan sesuatu yang lain.

Model asosiatif menyusun data menjadi dua set:

- Satu set item, masing-masing dengan pengidentifikasi unik, nama, dan tipe
- Satu set tautan, masing-masing dengan pengidentifikasi unik dan pengidentifikasi unik dari sumber, kata kerja, dan target. Fakta yang disimpan berkaitan dengan sumbernya,

dan masing-masing dari ketiga pengidentifikasi tersebut dapat merujuk ke tautan atau item.

Model database lain yang kurang umum meliputi:

- Model semantik, yang mencakup informasi tentang bagaimana data yang disimpan berhubungan dengan dunia nyata
- Basis data XML, yang memungkinkan data ditentukan dan bahkan disimpan dalam format XML
- Grafik bernama
- toko rangkap tiga

2.9 Model basis data NoSQL

Selain model database objek, model non-SQL lainnya juga muncul berbeda dengan model relasional:

Itu model basis data grafik, yang bahkan lebih fleksibel daripada model jaringan, memungkinkan setiap node terhubung dengan node lainnya. Itu model multinilai, yang keluar dari model relasional dengan mengizinkan atribut berisi daftar data, bukan satu titik data. Itu model dokumen, yang dirancang untuk menyimpan dan mengelola dokumen atau data semi-terstruktur, bukan data atom.

Basis Data di Web

Sebagian besar situs web mengandalkan beberapa jenis database untuk mengatur dan menyajikan data kepada pengguna. Setiap kali seseorang menggunakan fungsi pencarian di situs ini, istilah pencarian mereka diubah menjadi kueri untuk diproses oleh server database. Biasanya, middleware menghubungkan server web dengan database.

Kehadiran database yang luas memungkinkan database tersebut digunakan di hampir semua bidang, mulai dari belanja online hingga penargetan mikro pada segmen pemilih sebagai bagian dari kampanye politik. Berbagai industri telah mengembangkan norma mereka sendiri untuk desain database , mulai dari transportasi udara hingga manufaktur kendaraan.

Bab 3 Bahasa Pemrograman dalam Sistem Basis Data

3.1 Pengantar ke Bahasa Pemrograman Basis Data

Pengantar ke Bahasa Pemrograman Basis Data memperkenalkan pembaca pada berbagai bahasa pemrograman yang digunakan untuk mengakses dan memanipulasi data dalam sistem basis data. Bahasa pemrograman ini memungkinkan pengembang untuk membuat aplikasi yang dapat berinteraksi dengan database, melakukan operasi pengambilan data, penyisipan, pembaruan, dan penghapusan data, serta menjalankan prosedur dan fungsi yang terdefinisi dalam database. Salah satu bahasa pemrograman yang paling umum digunakan dalam konteks basis data adalah SQL (Structured Query Language). SQL adalah bahasa standar untuk mengakses dan mengelola data dalam database relasional. Dengan SQL, pengguna dapat membuat kueri untuk menarik informasi tertentu dari database, membuat, mengubah, atau menghapus tabel dan data, serta menjalankan operasi lainnya seperti pengaturan indeks dan hak akses pengguna.

Selain SQL, ada juga bahasa pemrograman yang disesuaikan dengan database tertentu, seperti PL/SQL untuk Oracle, T-SQL untuk SQL Server, dan PL/pgSQL untuk PostgreSQL. Bahasa-bahasa ini sering digunakan untuk membuat prosedur penyimpanan, fungsi, dan trigger yang berjalan di sisi server database untuk meningkatkan kinerja dan keamanan aplikasi. Selain itu, dengan perkembangan teknologi, sekarang ada juga kerangka kerja pemrograman yang memfasilitasi akses ke database, seperti ORM (Object-Relational Mapping), yang memungkinkan pengembang untuk berinteraksi dengan database menggunakan objek dan kelas dalam bahasa pemrograman yang lebih tinggi tingkatnya, seperti Java, Python, atau C#. Dengan pengenalan yang komprehensif terhadap bahasa pemrograman basis data, pembaca akan memiliki pemahaman yang kuat tentang bagaimana mengembangkan aplikasi yang kuat dan efisien yang berbasis pada sistem basis data, serta bagaimana memanfaatkan kemampuan database secara maksimal untuk mendukung kebutuhan bisnis dan teknis mereka.

Dalam istilah awam, database adalah kumpulan data dan informasi yang terorganisir. Basis data ini berisi banyak tabel dengan banyak bidang yang berisi informasi tentang klien atau karyawan perusahaan.

Hampir setiap situs web menggunakan database, terutama situs web eCommerce, untuk mencatat penjualan, pengiriman, dan informasi lainnya menggunakan perangkat lunak database khusus. Perangkat lunak ini penting untuk mengatur, memperbarui, dan menganalisis data, yang merupakan persyaratan situs web mana pun.

Ketika kita berbicara tentang perangkat lunak basis data, kita harus menyadari bahasa pemrograman yang digunakan pengembang untuk membuat perangkat lunak yang berharga ini. Gulir ke bawah untuk mempelajari tentang peran pemrograman basis data dan bahasa yang digunakan pemrogram dalam pemrograman basis data.

Apa itu Pemrograman Basis Data? Kalau frontend soal penampilan, backend soal performa. Hal ini ditentukan oleh jenis query database yang ditulis.

Bahasa database, biasa disebut bahasa query, adalah jenis bahasa pemrograman yang digunakan pemrogram untuk mendefinisikan dan mengakses data. Dalam sistem manajemen basis data (DBMS), bahasa-bahasa ini memungkinkan pengguna untuk melakukan tugas-tugas seperti:

- Membatasi akses data
- Mendefinisikan dan memodifikasi data
- Mencari informasi.

DBMS

Sistem manajemen basis data (DBMS) adalah sistem yang digunakan pengembang untuk mengelola basis data. SQL dan bahasa pemrograman database lainnya digunakan untuk menulis query untuk DBMS, seperti membuat, memodifikasi, dan menghapus record.

Meskipun database didefinisikan sebagai kumpulan tabel, tujuan utamanya bukan untuk menyimpan data. Misalnya, database yang dikelola dengan baik sangat penting bagi situs web eCommerce terkenal untuk menjaga minat kliennya. Jika terjadi kesalahan dan konsumen menerima pesanan yang berbeda, mereka tidak akan senang. Jika hal ini terjadi berulang kali, reputasi organisasi pasti akan terpuruk. Oleh karena itu, organisasi menggunakan perangkat lunak database khusus untuk mencegah kejadian buruk tersebut.

Perusahaan telah mulai merekrut administrator basis data yang memamerkan bakat mereka dalam pemrograman basis data untuk menghasilkan perangkat lunak basis data yang lebih efisien dan lebih cepat untuk diakses, diedit, dan diamankan, daripada memperoleh perangkat lunak basis data dasar.

Bahasa Pemrograman Basis Data

Bahasa pemrograman database adalah salah satu keterampilan paling menuntut di pasar kerja yang harus dikuasai seseorang. Berikut beberapa bahasa pemrograman database yang dapat Anda pelajari jika Anda ingin bekerja sebagai administrator atau manajer database. Kami akan melihat:

1. SQL
2. ular piton
3. PHP
4. C#
5. R

SQL

Structured Query Language (SQL) adalah salah satu bahasa paling mendasar yang harus dipahami jika Anda ingin bekerja dengan database. SQL adalah bahasa database yang dikembangkan oleh Oracle untuk membuat dan mengelola database.

SQL memungkinkan penyisipan, penghapusan, pengambilan, dan modifikasi data. Ini adalah bahasa database yang ideal untuk mengatur dan mengakses data secara efisien. Terlepas dari kekurangannya, SQL tetap menjadi salah satu bahasa database paling populer.

PYTHON

Python digunakan di hampir semua bidang, termasuk administrasi basis data. Misalnya, pengembang menggunakan python dalam manajemen basis data untuk menjalankan analisis data, tampilan, penyimpanan, dan fungsi relevan lainnya bila digabungkan dengan berbagai kerangka kerja.

PHP

PHP adalah bahasa skrip sisi server yang digunakan untuk mengembangkan halaman web dinamis yang menyimpan dan mengambil data secara bersamaan. Bukan suatu kebetulan jika PHP masuk dalam daftar tersebut, mengingat bahasa pemrograman ini memiliki hubungan khusus dengan database.

Pengembang dapat menggunakannya sebagai bahasa sisi server untuk membuat database canggih yang dapat memperbarui, menyimpan, dan mengambil data dengan cepat dan efisien.

C#

C# adalah bahasa pemrograman database lain yang mungkin menarik bagi administrator database. Meskipun C# didefinisikan sebagai bahasa pemrograman backend yang digunakan untuk membuat aplikasi lintas platform, C# juga memiliki keunggulan tambahan. Artinya, C# unggul dalam pengolahan data.

Ini adalah bahasa yang sesuai untuk kompatibilitas lintas platform dan pengembang dapat menerapkannya untuk database relasional guna memaksimalkan sumber daya sistem dan server.

Tidak seperti bahasa lain, seperti SQL, yang sebagian besar digunakan oleh pengembang untuk pengelolaan dan pengambilan data, pemrogram hanya menggunakan C# untuk pemrosesan dan tidak dapat melakukan tugas yang diselesaikan SQL. Jadi, mempelajari dua bahasa bermanfaat bagi programmer.

R

Peran administrator data adalah mengelola database, menjalankan berbagai operasi, dan memberikan laporan. Dan itu memakan waktu. Belum lagi fakta bahwa organisasi harus mengelola database besar yang berisi ribuan catatan. Dan di sinilah R datang untuk menyelamatkan.

R adalah bahasa pemrograman yang dirancang untuk menangani tugas-tugas rumit, termasuk penambangan data, analisis data, komputasi statistik, dan banyak lagi.

R dapat menangani operasi yang memerlukan persamaan matematika untuk menampilkan, menganalisis, dan menyimpan informasi saat bekerja dengan data yang kompleks dan persyaratan pemrosesan.

3.2 SQL (Structured Query Language)

Structured Query Language (SQL) mengacu pada bahasa pemrograman standar yang digunakan untuk mengekstrak, mengatur, mengelola, dan memanipulasi data yang disimpan dalam database relasional. SQL dengan demikian disebut sebagai bahasa database yang dapat menjalankan aktivitas pada database yang terdiri dari tabel yang terdiri dari baris dan kolom.

SQL memainkan peran penting dalam mengambil data yang relevan dari database, yang nantinya dapat digunakan oleh berbagai platform seperti Python atau R untuk tujuan analisis. SQL dapat mengelola beberapa transaksi data secara bersamaan dimana sejumlah besar data ditulis secara bersamaan.

SQL adalah standar American National Standards Institute (ANSI) yang beroperasi melalui berbagai versi dan kerangka kerja untuk menangani data backend di berbagai aplikasi web yang didukung oleh database relasional seperti MySQL, SQL Server, Oracle PostgreSQL , dan lainnya.

Perusahaan-perusahaan terkemuka yang dimiliki oleh Meta Inc., seperti Facebook, WhatsApp, dan Instagram , semuanya mengandalkan SQL untuk pemrosesan data dan penyimpanan backend.

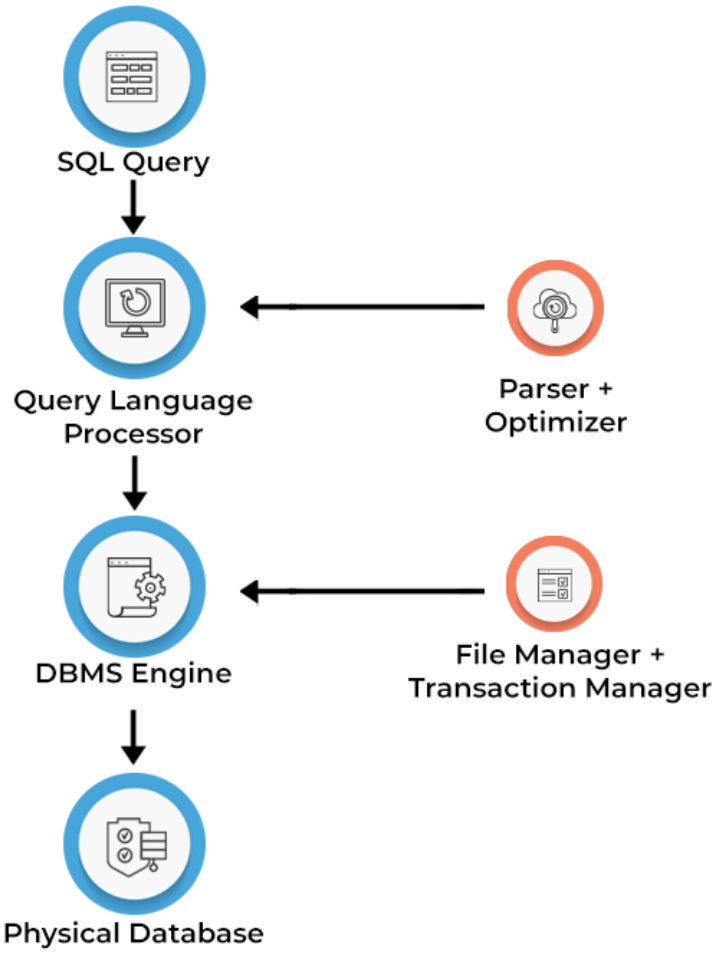
Bagaimana cara kerja SQL?

Saat kueri SQL ditulis dan dijalankan, kueri tersebut diproses oleh 'pemroses bahasa kueri' yang memiliki parser dan pengoptimal kueri. Server SQL kemudian mengkompilasi kueri yang diproses dalam tiga tahap:

1. Parsing: Ini mengacu pada proses yang memeriksa ulang sintaks kueri.

2. Binding: Langkah ini melibatkan verifikasi semantik kueri sebelum menjalankannya.

3. Optimasi: Langkah terakhir menghasilkan rencana eksekusi query. Tujuannya di sini adalah untuk mengidentifikasi rencana eksekusi kueri efisien yang berjalan dalam waktu minimal. Artinya semakin pendek waktu respon query SQL, semakin baik hasilnya. Beberapa kombinasi rencana dihasilkan untuk mendapatkan rencana pelaksanaan akhir yang praktis.



Gambar 3. 1 Contoh Cara Kerja SQL

Manfaat SQL penting

SQL menawarkan beberapa keuntungan karena merupakan bahasa yang ramah pengguna dan dapat diakses di seluruh platform.

Mari kita pahami manfaat utama SQL :

- Bahasa portabel: SQL, sebagai bahasa portabel, dapat ditransfer dari satu perangkat ke perangkat lainnya, di mana perangkat tersebut dapat berkisar dari komputer pribadi dan server hingga laptop dan bahkan beberapa perangkat seluler. Bahasa ini mampu berjalan di internet lokal dan sistem intranet.
- Pemrosesan kueri yang cepat: Berapa pun volume datanya, SQL mampu menyisipkan, menghapus, mengambil, dan memanipulasi data dengan cepat dan efisien sekaligus memastikan keakuratan data. Hal ini memungkinkan berbagi data dengan cepat antar pengguna.
- Tidak diperlukan keahlian coding: SQL tidak menuntut keahlian coding seperti bahasa pemrograman lainnya. Sifatnya yang mudah digunakan membuatnya dapat diakses oleh semua pengguna karena mereka dapat mengelola SQL dengan bantuan kata kunci seperti 'buat', 'masukkan', 'pilih', 'perbarui', dan lainnya tanpa memiliki keahlian pemrograman apa pun.
- Platform seragam dengan bahasa standar : SQL menggunakan bahasa Inggris sebagai bahasa standar; karenanya, mudah bagi semua pengguna untuk memahami, mempelajari, menulis, dan menafsirkan tanpa banyak kesulitan. Kata-kata dan pernyataan dalam bahasa Inggris membuat SLQ dapat diakses oleh semua orang, termasuk orang-orang dengan sedikit atau tanpa pengalaman sebelumnya.
- Menawarkan beberapa tampilan data: SQL menyediakan fasilitas untuk membuat beberapa tampilan data, di mana pengguna yang berbeda dapat memvisualisasikan struktur database dan kontennya secara berbeda.
- Kode sumber terbuka . Solusi SQL sumber terbuka seperti MySQL, MariaDB, dan PostGresSQL menyediakan database SQL yang dapat diakses. Hal ini menarik partisipasi masyarakat yang lebih besar dengan biaya yang lebih rendah.
- Vendor sistem manajemen basis data (DBMS) teratas menggunakan SQL : Sistem DBMS dari perusahaan terkemuka seperti IBM, Oracle, dan Microsoft menggunakan SQL, dengan mempertimbangkan manfaat komprehensif yang ditawarkannya.
- Bahasa interaktif : SQL adalah bahasa interaktif dan interpretatif. Dengan demikian, ini mengurangi kemungkinan miskomunikasi atau kesalahpahaman antar pengguna.

Elemen SQL

SQL adalah pilihan utama sebagian besar pengguna database karena kemudahan penggunaannya dan bagaimana kueri dapat menjalankan beragam fungsi pada data terstruktur dalam jumlah besar.

Bahasa pemrograman SQL memiliki elemen penting berikut:

1. Keywords

2. Clauses

3. Expressions

4. Predicates

5. Queries

Gambar 3. 2 Contoh Elemen SQL

Mari kita pahami peran setiap elemen dalam pemrograman SQL:

Keywords: Kata kunci mengacu pada sekumpulan kata yang memungkinkan Anda melakukan operasi pada database Anda. Perhatikan contoh kata kunci ' LIKE '; itu mencari pola data tertentu dalam database.

Misalnya, kita perlu mengidentifikasi nama keluarga yang tinggal di wilayah Boston yang memiliki nama belakang 'Luis'. Kueri SQL berikut akan mengambil hasil yang relevan untuk pernyataan masalah ini:

PILIH * DARI [BOSTON] DI MANA NAMA SEPERTI '%LUIS'

Demikian pula, kata kunci yang berbeda melakukan berbagai operasi pada database. Berikut ini adalah beberapa contoh kata kunci tersebut beserta peran fungsionalnya:

- BUAT: Kata kunci ini membantu dalam membuat struktur database atau sekadar tabel, tampilan, dan indeks
- INSERT: Ini menambahkan data ke baris tabel

- SELECT: Memilih data dari database atau tabel
- FROM: Menunjukkan tabel dari mana data perlu diambil
- WHERE: Ini memfilter data sehingga hanya data relevan yang cocok dengan kondisi tertentu yang diambil
- PEMBARUAN: Memperbarui baris yang ada dalam tabel
- DELETE: Menghapus baris-baris yang ada pada tabel

Clauses: Klausul mengacu pada fungsi bawaan yang menyaring data dan mengambil data yang diperlukan dari database atau tabel. Sangat cocok ketika menangani database besar. Klausul adalah bagian dari pernyataan SQL.

Mari pertimbangkan kasus penggunaan di mana Anda perlu memilih usia, email, dan alamat dari database. Seseorang kemudian akan mewakili klausa sebagai 'SELECT Age, Email, and Address,' di mana SELECT adalah kata kunci dan age, email, dan alamat mengungkapkan informasi tertentu untuk menjalankan query SQL dan mengambil data yang diperlukan.

Expressions: Ekspresi SQL mewakili rumus yang biasanya ditulis dalam format kueri. Ini menggabungkan satu atau lebih nilai, operator, dan fungsi SQL yang mengevaluasi nilai tertentu. Selain itu, ekspresi SQL secara garis besar dibagi menjadi tiga jenis, yaitu Boolean, numerik, dan tanggal .

Mari kita perhatikan contoh ekspresi Boolean yang mengambil data dengan mencocokkan nilai tunggal. Jika Anda ingin mengidentifikasi karyawan yang gajinya sama dengan 5.000, seseorang dapat menggunakan kueri SQL berikut:

PILIH*DARI KARYAWAN DIMANA GAJI = 5000;

Predicates: Predikat mengacu pada kata kunci yang mengungkapkan hubungan antara dua ekspresi dan menghasilkan nilai benar atau salah. Ini hanyalah istilah lain untuk ekspresi yang digunakan untuk menentukan kondisi yang tidak diketahui atau kondisi TRUE/FALSE.

Misalnya, pertimbangkan pernyataan SQL berikut:

PILIH * DARI PELANGGAN DI MANA Produk = 'Televisi' ;

Di sini, 'Produk = Televisi' adalah predikat pernyataan SQL.

Queries: Kueri SQL mengacu pada pernyataan yang digunakan untuk meminta atau mengambil data dari database. Misalnya, Anda ingin mengambil nama depan dan nomor pelanggan dari semua pelanggan yang nama belakangnya 'Lobo'.

Contoh SQL

SQL menanyakan database relasional untuk mengambil data yang direpresentasikan dalam format tabel (baris dan kolom). Di sini, baris melambangkan objek yang berbeda, sedangkan kolom melambangkan atribut dari objek tersebut.

Mari kita pahami beberapa contoh dasar bahasa pemrograman SQL .

Mari kita buat tabel yang menyimpan data cuaca untuk berbagai kota dan tidak memiliki kolom ID duplikat.

BUAT CUACA TABEL

(ID KUNCI UTAMA INTEGER,

CITY_US CHAR (30),

STATE_US CHAR(30),

LATITUDE_N NYATA,

LONGITUDE_W NYATA);

Sekarang setelah tabel dibuat di backend, mari kita isi tabel CUACA dengan data:

MASUKKAN KE NILAI CUACA (19, 'Los Angeles,' 'CA,' 34, 118);

MASUKKAN KE NILAI CUACA (38, Detroit, 'MI,' 42, 83);

MASUKKAN KE NILAI CUACA (97, Spokane, 'WA,' 47, 117);

Setelah mengisi tabel, Anda dapat melihat tabel CUACA dengan kueri SQL berikut dalam urutan yang tidak ditentukan:

PILIH * DARI CUACA;

PENGENAL	CITY_US	STATE_US	LATITUDE_N	LONGITUDE_W
19	Los Angeles	CA	34	118
38	Detroit	MI	42	83
97	Spokane	WA	47	117

Gambar 3. 3 Contoh Data Cuaca

Untuk melihat kota dan negara bagian yang garis lintangnya > 40, gunakan kueri berikut:

PILIH * DARI CUACA DI MANA LATITUDE_N > 40;

PENGENAL	CITY_US	STATE_US	LATITUDE_N	LONGITUDE_W
38	Detroit	MI	42	83
97	Spokane	WA	47	117

Gambar 3. 4 Contoh Letak Cuaca

Pemilihan baris seperti ini merupakan cara mendasar untuk menerapkan 'pembatasan' pada data yang akan dilihat. Mari kita lihat contoh yang memungkinkan Anda memilih hanya kolom tertentu dari tabel asli, seperti kolom ID, CITY_US, dan LATITUDE_N :

PILIH ID, CITY_US, LATITUDE_N DARI CUACA;

PENGENAL	CITY_US	LATITUDE_N
19	Los Angeles	34
38	Detroit	42
97	Spokane	47

Gambar 3. 5 Contoh ID Cuaca Tabel Asli

Pilihan kolom tertentu disebut sebagai 'proyeksi' dalam terminologi SQL. Sekarang mari kita gabungkan kasus penggunaan 'batasi' dan 'proyek' dalam satu kueri SQL:

PILIH ID, CITY_US, STATE_US DARI CUACA DI MANA LATITUDE_N > 40;

PENGENAL	CITY_US	STATE_US
38	Detroit	MI
97	Spokane	WA

Gambar 3. 6 Contoh Penggabungan Kasus

Ini adalah beberapa **contoh SQL interaktif** yang dapat digunakan untuk mengutak-atik model database relasional.

3.3 Penggunaan SQL

SQL adalah bahasa khusus domain populer yang digunakan untuk bertransaksi dengan database. Itu dapat melakukan beberapa operasi pada database atau tabel, seperti mengekstraksi, memodifikasi, dan menyimpan data, dan banyak lainnya. Sekarang mari kita pahami beberapa penggunaan penting SQL:

Pengelolaan database relasional

Sejarah database relasional dimulai pada tahun 1969 ketika ilmuwan Edgar F. Codd pertama kali mendefinisikannya. Kemudian pada tahun 1970 dikembangkan perangkat lunak untuk memelihara model relasional dengan menggunakan bahasa SQL. Kode ini disebut sebagai Sistem Manajemen Basis Data Relasional (RDBMS) . Dalam database relasional, data disimpan dalam baris. Ini disebut 'tupel' dan dikelompokkan berdasarkan hubungan umum. Jadi, penggunaan SQL yang paling mendasar adalah untuk memelihara data di RDBMS dan melakukan operasi pada data tersebut dengan menggunakan kueri SQL.

Melakukan operasi dasar database

SQL menggunakan beberapa perintah utama saat berinteraksi dengan database. Ini termasuk:

- Bahasa Definisi Data (DDL) : SQL menggunakan perintah untuk membuat dan memelihara database seperti CREATE, DROP, ALTER, TRUNCATE, dan COMMENT.
- Data Query Language (DQL) : Ini mengacu pada perintah SQL yang digunakan untuk mengambil data dari database. Perintah SQL yang umum digunakan di sini adalah SELECT.
- Bahasa Manipulasi Data (DML) : Ini mengacu pada perintah SQL yang digunakan untuk memanipulasi data dan melakukan operasi penting pada database, seperti INSERT, UPDATE, dan DELETE.
- Bahasa Kontrol Data (DCL) : Ini mengacu pada perintah kontrol SQL yang memberikan izin atau akses kepada pengguna untuk melakukan operasi tertentu. Misalnya REVOKE adalah perintah kontrol yang mencabut izin akses yang diberikan kepada pengguna.

Mengizinkan kontrol transaksi database

SQL digunakan untuk menjaga konsistensi dalam transaksi database. Ini mengikuti aturan dasar ACID, di mana ACID mengacu pada atomisitas, konsistensi, isolasi, dan daya tahan . Pemrosesan transaksi dilakukan jika mematuhi semua sifat dasar ini.

Atomicity menyiratkan bahwa setiap transaksi adalah unit atom, dan semua instruksi terjadi dengan sukses, atau tidak ada satupun yang dijalankan sama sekali. Konsistensi berarti database harus konsisten sebelum dan sesudah setiap transaksi. Isolasi menyiratkan bahwa beberapa transaksi yang berjalan secara bersamaan tidak boleh saling mempengaruhi dengan cara apa pun. Terakhir, ketahanan menunjukkan bahwa jika terjadi kegagalan perangkat lunak atau sistem, perubahan yang dilakukan pada data harus tetap ada setelah penyelesaian transaksi.

Eksekusi perintah SQL UNION dan JOIN

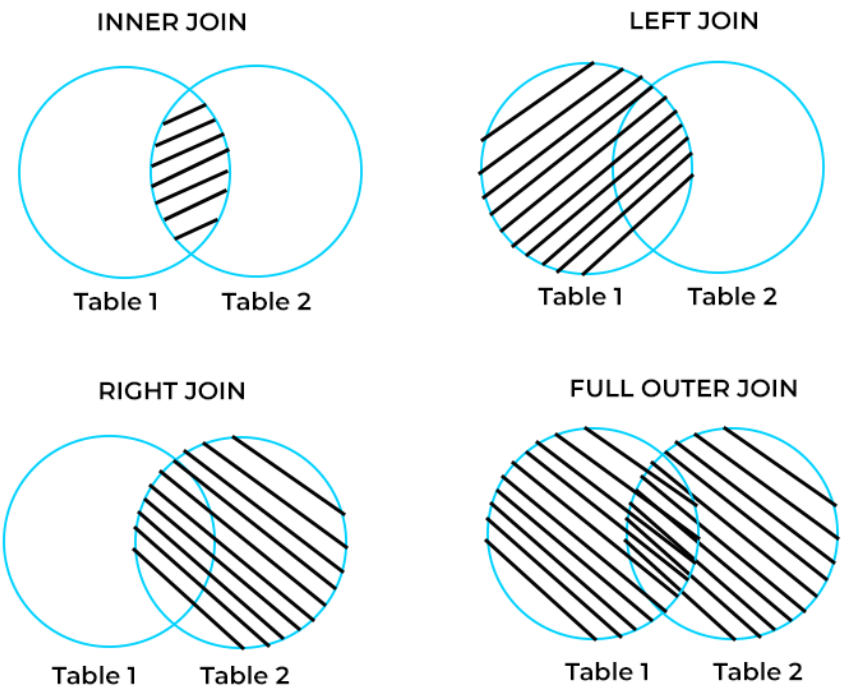
SQL menggunakan perintah UNION dan JOIN untuk menggabungkan dua tabel dalam database untuk memberikan satu hasil atau keluaran. Mari kita pertimbangkan skenario ketika Anda ingin melihat informasi pelanggan yang membeli produk tertentu dari mal. Detail pelanggan ada di tabel 'pelanggan', sedangkan detail produk ada di tabel 'produk'. Oleh karena itu, untuk melihat informasi pelanggan tentang siapa yang membeli produk tertentu, seseorang dapat melakukan operasi GABUNG.

JOIN dibagi lagi menjadi empat jenis:

- GABUNG DALAM
- KIRI (LUAR) GABUNG
- GABUNG KANAN (LUAR).

- GABUNG LENGKAP (LUAR).

Semua operasi GABUNG ini direpresentasikan seperti pada gambar berikut:



Gambar 3. 7 Contoh Join

SQL untuk situs web

Setiap situs web memiliki database backend yang berisi banyak sekali data (pengguna, produk, dll.) yang perlu diambil jika diperlukan. Saat ini, sebagian besar situs web interaktif cenderung menggunakan bahasa kueri SQL untuk mengekstrak dan menyimpan data dalam database. Misalnya, situs e-niaga seperti Amazon dan Flipkart menggunakan MySQL untuk pengelolaan data yang mendukung SQL standar. Selain itu, situs pemesanan film dan akomodasi juga mengandalkan bahasa kueri untuk melakukan operasi pada data sesuai kebutuhan.

Kompatibilitas dan dukungan perpustakaan

Perusahaan seperti Amazon, Google, Microsoft, Oracle**,** dan lainnya menggunakan database relasional seperti Microsoft SQL Server, MS Access, database Oracle, MYSQL, dll. SQL biasanya kompatibel dengan sebagian besar database relasional tersebut.

Selain itu, SQL menawarkan kumpulan perpustakaan, seperti SQLite, yang membantu menghubungkan aplikasi web klien ke pengembang database. Dukungan perpustakaan SQL memungkinkan pengguna untuk bekerja dengan kumpulan data klien.

Integrasi dengan bahasa scripting lainnya

SQL terintegrasi dengan bahasa skrip populer seperti Python dan R. Hal ini memungkinkan SQL untuk mengelola database yang didukung oleh bahasa-bahasa ini, dan ini sangat berguna bagi pengguna seperti analis data, ilmuwan data, teknisi ML, dll., yang cenderung menangani data dalam jumlah besar secara bersamaan.

SQL adalah bahasa kueri fungsional karena menggunakan kueri sederhana untuk mengelola kumpulan data besar. Selain itu, operasi kompleks seperti pemfilteran data, pemotongan, pengindeksan, dll., dapat dilakukan dengan menggunakan klausa sederhana seperti WHERE dan lainnya dengan operator logika dan kondisi dalam kueri SQL.

SQL untuk pembelajaran mesin

Pembelajaran mesin muncul ketika Anda ingin menangani data dalam jumlah besar. Selain itu, di mana pun terdapat kumpulan data berukuran besar, SQL pasti akan menemukan tempatnya dalam situasi seperti itu.

Platform cloud Google BigQuery adalah contoh bagus SQL untuk ML. Platform ini menggunakan kueri dan alat SQL untuk membuat dan menjalankan berbagai model ML. Dengan kata sederhana, model ML dipanggil pada data terstruktur dengan menggunakan SQL. Oleh karena itu, dukungan ML SQL dan integrasinya yang mudah dengan bahasa skrip menjadikannya bahasa kueri yang cocok untuk aplikasi yang menangani data dalam jumlah besar.

Bab 4 Desain Basis Data

4.1 Proses Desain Basis Data

Prinsip-prinsip tertentu memandu proses desain database. Prinsip pertama adalah bahwa informasi duplikat (juga disebut data berlebihan) buruk, karena itu membuang ruang dan meningkatkan kemungkinan kesalahan dan inkonsistensi. Prinsip kedua adalah bahwa kebenaran dan kelengkapan informasi adalah penting. Jika database Anda berisi informasi yang salah, setiap laporan yang mengambil informasi dari database juga akan berisi informasi yang salah. Hasilnya, keputusan apa pun yang Anda buat berdasarkan laporan tersebut kemudian akan salah disalahartikan.

Oleh karena itu, desain database yang bagus adalah:

- Membagi informasi Anda menjadi tabel berbasis subjek untuk mengurangi data yang berlebihan.
- Menyediakan Access dengan informasi yang diperlukan untuk menggabungkan informasi dalam tabel bersama-sama sesuai kebutuhan.
- Membantu mendukung dan memastikan keakuratan dan integritas informasi Anda.
- Mengakomodasi kebutuhan pemrosesan dan pelaporan data Anda.

Proses desain

Proses desain terdiri dari langkah-langkah berikut:

- Menentukan tujuan database Anda
  Ini membantu mempersiapkan Anda untuk langkah-langkah yang tersisa.

- Menemukan dan menata informasi yang diperlukan
  Kumpulkan semua tipe informasi yang mungkin ingin Anda rekam dalam database, seperti nama produk dan nomor pesanan.

- Membagi informasi ke dalam tabel
  Bagi item informasi Anda menjadi entitas atau subjek utama, seperti Produk atau Pesanan. Setiap subjek kemudian menjadi tabel.

- Mengubah item informasi menjadi kolom

Tentukan informasi apa yang ingin Anda simpan di setiap tabel. Setiap item menjadi bidang, dan ditampilkan sebagai kolom dalam tabel. Misalnya, tabel Karyawan mungkin menyertakan bidang seperti Nama Belakang dan Tanggal Sewa.

- Menentukan kunci utama
  Pilih kunci utama setiap tabel. Kunci utama adalah kolom yang digunakan untuk mengidentifikasi setiap baris secara unik. Contohnya mungkin ID Produk atau ID Pesanan.

- Menyiapkan hubungan tabel
  Lihat setiap tabel dan putuskan bagaimana data dalam satu tabel terkait dengan data dalam tabel lain. Tambahkan bidang ke tabel atau buat tabel baru untuk mengklarifikasi hubungan, jika diperlukan.

- Persempit desain Anda
  Menganalisis desain Anda untuk kesalahan. Buat tabel dan tambahkan beberapa rekaman data sampel. Lihat apakah Anda bisa mendapatkan hasil yang Anda inginkan dari tabel Anda. Buat penyesuaian pada desain, sesuai kebutuhan.

- Menerapkan aturan normalisasi
  Terapkan aturan normalisasi data untuk melihat apakah tabel Anda disusun dengan benar. Buat penyesuaian pada tabel, sesuai kebutuhan.

Menentukan tujuan database Anda

Merupakan ide yang baik untuk menuliskan tujuan database di kertas — tujuannya, bagaimana Anda berharap untuk menggunakannya, dan siapa yang akan menggunakannya. Untuk database kecil untuk bisnis berbasis rumahan, misalnya, Anda mungkin menulis sesuatu yang sederhana seperti "Database pelanggan menyimpan daftar informasi pelanggan untuk tujuan memproduksi surat dan laporan." Jika database lebih kompleks atau digunakan oleh banyak orang, seperti yang sering terjadi dalam pengaturan perusahaan, tujuannya bisa dengan mudah berupa paragraf atau lebih dan harus menyertakan kapan dan bagaimana setiap orang akan menggunakan database. Idenya adalah memiliki pernyataan misi yang dikembangkan dengan baik yang dapat dirujuk sepanjang proses desain. Memiliki pernyataan seperti itu membantu Anda fokus pada tujuan Anda ketika Anda membuat keputusan.

Menemukan dan menata informasi yang diperlukan

Untuk menemukan dan menata informasi yang diperlukan, mulailah dengan informasi Anda yang sudah ada. Misalnya, Anda dapat merekam pesanan pembelian dalam buku besar atau menyimpan informasi pelanggan pada formulir kertas dalam lemari file. Kumpulkan dokumen

tersebut dan cantumkan setiap tipe informasi yang diperlihatkan (misalnya, setiap kotak yang Anda isi pada formulir). Jika Anda tidak memiliki formulir yang sudah ada, bayangkan Anda harus mendesain formulir untuk merekam informasi pelanggan. Informasi apa yang akan Anda masukkan pada formulir? Kotak isian apa yang akan Anda buat? Identifikasi dan cantumkan setiap item ini. Misalnya, anggaplah Anda saat ini menyimpan daftar pelanggan di kartu indeks. Memeriksa kartu ini mungkin menunjukkan bahwa setiap kartu berisi nama, alamat, kota, negara bagian, kode pos, dan nomor telepon pelanggan. Setiap item ini mewakili kolom potensial dalam tabel.

Saat Anda mempersiapkan daftar ini, jangan khawatir tentang menyempurnakannya pada awalnya. Sebagai gantinya, cantumkan setiap item yang muncul dalam pikiran. Jika orang lain akan menggunakan database, tanyakan juga ide mereka. Anda dapat menyempurnakan daftar nanti.

Berikutnya, pertimbangkan tipe laporan atau surat yang mungkin ingin Anda hasilkan dari database. Misalnya, Anda mungkin ingin laporan penjualan produk memperlihatkan penjualan menurut kawasan, atau laporan ringkasan inventarisi yang memperlihatkan tingkat inventarsi produk. Anda mungkin juga ingin membuat surat formulir untuk dikirimkan kepada pelanggan yang membacakan acara penjualan atau menawarkan premium. Desain laporan dalam pikiran Anda, dan bayangkan seperti apa tampilannya. Informasi apa yang akan Anda tempatkan pada laporan? Mencantumkan setiap item. Lakukan hal yang sama untuk surat formulir dan untuk laporan lain yang Anda antisipasi untuk dibuat.

Gambar 4. 1 Contoh Desain Laporan

Memberikan pemikiran pada laporan dan surat yang mungkin ingin Anda buat membantu Anda mengidentifikasi item yang akan Anda perlukan dalam database Anda. Misalnya, anggaplah Anda memberi pelanggan kesempatan untuk ikut serta dalam (atau tidak) pembaruan email berkala, dan Anda ingin mencetak daftar orang-orang yang telah ikut serta. Untuk merekam informasi tersebut, Tambahkan kolom "Kirim email" ke tabel pelanggan. Untuk setiap pelanggan, Anda dapat mengatur bidang ke Ya atau Tidak.

Persyaratan untuk mengirim pesan email ke pelanggan menyarankan item lain untuk direkam. Setelah mengetahui bahwa pelanggan ingin menerima pesan email, Anda juga perlu mengetahui alamat email yang akan dikirimkan kepada mereka. Oleh karena itu Anda perlu merekam alamat email untuk setiap pelanggan.

Masuk akal untuk membuat prototipe dari setiap laporan atau daftar output dan mempertimbangkan item apa yang akan Anda perlukan untuk menghasilkan laporan. Misalnya, ketika Anda memeriksa surat formulir, beberapa hal mungkin muncul dalam pikiran Anda. Jika Anda ingin menyertakan salam yang tepat — misalnya, string "Mr.", "Mrs." atau "Ms." yang memulai salam, Anda harus membuat item salam. Juga, Anda mungkin biasanya memulai surat dengan "Dear Mr Smith", bukan "Dear. Tn. Sylvester Smith". Hal ini menunjukkan bahwa Anda biasanya ingin menyimpan nama belakang yang terpisah dari nama depan.

Poin penting untuk diingat adalah bahwa Anda harus memecah setiap bagian informasi ke dalam bagian-bagian terkecil yang berguna. Dalam kasus nama, untuk membuat nama belakang tersedia dengan mudah, Anda akan memecah nama menjadi dua bagian — Nama Depan dan Nama Belakang. Untuk mengurutkan laporan menurut nama belakang, misalnya, membantu menyimpan nama belakang pelanggan secara terpisah. Secara umum, jika Anda ingin mengurutkan, mencari, menghitung, atau melaporkan berdasarkan item informasi, Anda harus meletakkan item itu di bidangnya sendiri.

Pikirkan tentang pertanyaan yang mungkin Anda inginkan untuk dijawab oleh database. Misalnya, berapa banyak penjualan produk unggulan Anda yang Anda tutup bulan lalu? Di mana pelanggan terbaik Anda tinggal? Siapa pemasok untuk produk terlaris Anda? Mengantisipasi pertanyaan ini membantu Anda mendata item tambahan.

Membadingkan informasi ke dalam tabel

Untuk membagi informasi menjadi tabel, pilih entitas utama, atau subjek. Misalnya, setelah menemukan dan menata informasi untuk database penjualan produk, daftar pendahuluan mungkin terlihat seperti ini:

Gambar 4. 2 Contoh Tabel Informasi

Entitas utama yang ditunjukkan di sini adalah produk, pemasok, pelanggan, dan pesanan. Oleh karena itu, masuk akal untuk memulai dengan empat tabel ini: satu untuk fakta tentang produk, satu untuk fakta tentang pemasok, satu untuk fakta tentang pelanggan, dan satu untuk fakta tentang pesanan. Meskipun ini tidak melengkapi daftar, ini adalah titik awal yang baik. Anda bisa terus memperbaiki daftar ini hingga Anda memiliki desain yang berfungsi dengan baik.

Saat pertama kali meninjau daftar item pendahuluan, Anda mungkin tergoda untuk menempatkan semuanya dalam satu tabel, bukan empat item yang diperlihatkan dalam ilustrasi sebelumnya. Anda akan belajar di sini mengapa itu adalah ide yang buruk. Pertimbangkan untuk sesaat, tabel yang diperlihatkan di sini:

Produk dan Pemasok

ProductName	Pemasok	Alamat
Chai	Eksotis cairan	49 St Gilbert
Chang	eksotis cairan	49 St. Gilbert.
Asdteed sirup	Eksotis cairan	49Kotak Gilbert
Chef Anton Cajun laut	New Orleans Cajun Deli	St. P.O. 78934É

Gambar 4. 3 Contoh Tabel Database

Dalam hal ini, setiap baris berisi informasi tentang produk dan pemasoknya. Karena Anda dapat memiliki banyak produk dari pemasok yang sama, nama pemasok dan informasi alamat harus diulang berkali-kali. Ini membuang ruang disk. Merekam informasi pemasok hanya sekali

dalam tabel Pemasok terpisah, lalu menautkan tabel tersebut ke tabel Produk, adalah solusi yang jauh lebih baik.

Masalah kedua dengan desain ini muncul ketika Anda perlu mengubah informasi tentang pemasok. Misalnya, Anda perlu mengubah alamat pemasok. Karena alamat muncul di banyak tempat, Anda mungkin secara tidak sengaja mengubah alamat di satu tempat tetapi lupa untuk mengubahnya di tempat lain. Merekam alamat pemasok hanya di satu tempat yang menyelesaikan masalah.

Saat Anda mendesain database Anda, selalu coba merekam setiap fakta sekali saja. Jika Anda menemukan diri Anda mengulangi informasi yang sama di lebih dari satu tempat, seperti alamat untuk pemasok tertentu, tempatkan informasi tersebut dalam tabel terpisah.

Akhirnya, misalkan hanya ada satu produk yang disediakan oleh Coho Winery, dan Anda ingin menghapus produk, tetapi mempertahankan nama pemasok dan informasi alamat. Bagaimana Anda menghapus catatan produk tanpa kehilangan informasi pemasok juga? Anda tidak bisa. Karena setiap catatan berisi fakta tentang produk, serta fakta tentang pemasok, Anda tidak dapat menghapusnya tanpa menghapus yang lain. Untuk memisahkan fakta-fakta ini, Anda harus membagi satu tabel menjadi dua: satu tabel untuk informasi produk, dan tabel lain untuk informasi pemasok. Menghapus catatan produk harus menghapus hanya fakta tentang produk, bukan fakta tentang pemasok.

Setelah Memilih subjek yang dinyatakan oleh tabel, kolom dalam tabel tersebut hanya akan menyimpan fakta tentang subjek. Misalnya, tabel produk harus menyimpan fakta hanya tentang produk. Karena alamat pemasok adalah fakta tentang pemasok, dan bukan fakta tentang produk, itu termasuk dalam tabel pemasok.

Mengubah item informasi menjadi kolom

Untuk menentukan kolom dalam tabel, tentukan informasi apa yang perlu Anda lacak tentang subjek yang direkam dalam tabel. Misalnya, untuk tabel Pelanggan, Nama, Alamat, Kota-Negara-Zip, Kirim email, Salam dan Alamat email terdiri dari daftar awal kolom yang baik. Setiap catatan dalam tabel berisi kumpulan kolom yang sama, sehingga Anda bisa menyimpan informasi Nama, Alamat, Kota-Negara-Zip, Kirim email, Salam dan Alamat email untuk setiap catatan. Misalnya, kolom alamat berisi alamat pelanggan. Setiap catatan berisi data tentang satu pelanggan, dan bidang alamat berisi alamat untuk pelanggan tersebut.

Setelah menentukan kumpulan kolom awal untuk setiap tabel, Anda dapat lebih mempersempit kolom. Misalnya, masuk akal untuk menyimpan nama pelanggan sebagai dua kolom terpisah: nama depan dan nama belakang, sehingga Anda bisa mengurutkan, mencari, dan mengindeks hanya pada kolom tersebut. Demikian pula, alamat sebenarnya terdiri dari lima komponen, alamat, kota, negara bagian, kode pos, dan negara/kawasan terpisah, dan juga masuk akal untuk menyimpannya di kolom terpisah. Jika Anda ingin melakukan operasi pencarian,

filter, atau pengurutan menurut status, misalnya, Anda memerlukan informasi status yang disimpan dalam kolom terpisah.

Anda juga harus mempertimbangkan apakah database akan menyimpan informasi yang berasal dari dalam negeri saja, atau internasional, juga. Misalnya, jika Anda berencana untuk menyimpan alamat internasional, lebih baik memiliki kolom Kawasan dan bukan Negara Bagian, karena kolom tersebut dapat mengakomodasi negara bagian domestik dan kawasan negara/kawasan lain. Demikian pula, Kode Pos lebih masuk akal daripada Kode Pos jika Anda akan menyimpan alamat internasional.

Daftar berikut ini memperlihatkan beberapa tips untuk menentukan kolom Anda.

- Jangan sertakan data terhitung

  Biasanya, Anda tidak boleh menyimpan hasil penghitungan dalam tabel. Sebagai gantinya, Anda bisa meminta Access melakukan penghitungan saat Anda ingin melihat hasilnya. Misalnya, anggaplah ada laporan Produk Sesuai Pesanan yang menampilkan subtotal unit sesuai urutan untuk setiap kategori produk dalam database. Namun, tidak ada kolom subtotal Unit Sesuai Pesanan dalam tabel apa pun. Sebagai gantinya, tabel Produk menyertakan kolom Unit Sesuai Pesanan yang menyimpan unit sesuai pesanan untuk setiap produk. Dengan data tersebut, Access menghitung subtotal setiap kali Anda mencetak laporan. Subtotal itu sendiri tidak boleh disimpan dalam tabel.

- Menyimpan informasi di bagian logika terkecilnya

  Anda mungkin tergoda untuk memiliki satu bidang untuk nama lengkap, atau untuk nama produk beserta deskripsi produk. Jika Anda menggabungkan lebih dari satu jenis informasi dalam bidang, sulit untuk mengambil fakta individual nanti. Cobalah untuk memecah informasi menjadi bagian logika; misalnya, buat bidang terpisah untuk nama depan dan belakang, atau untuk nama produk, kategori, dan deskripsi.

Gambar 4. 4 Contoh Hasil Mengubah Item Informasi Menjadi Kolom

Setelah anda telah menyempurnakan kolom data di setiap tabel, Anda siap untuk memilih setiap kunci utama tabel.

Menentukan kunci utama

Setiap tabel harus menyertakan kolom atau sekumpulan kolom yang mengidentifikasi setiap baris yang disimpan dalam tabel secara unik. Ini seringkali merupakan nomor identifikasi unik, seperti nomor ID karyawan atau nomor seri. Dalam terminologi database, informasi ini disebut kunci utama tabel. Access menggunakan bidang kunci utama untuk mengaitkan data dengan cepat dari beberapa tabel dan menyatukan data untuk Anda.

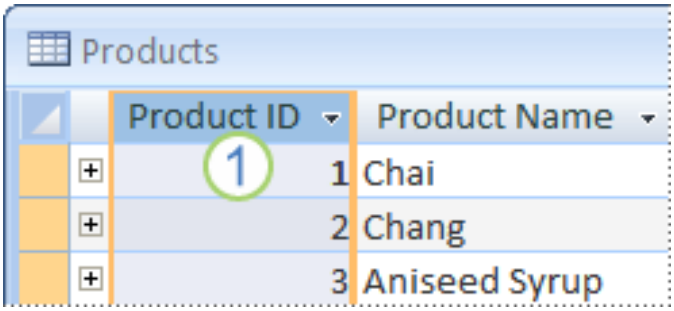
Jika Anda sudah memiliki pengidentifikasi unik untuk tabel, seperti nomor produk yang mengidentifikasi setiap produk secara unik dalam katalog, Anda bisa menggunakan pengidentifikasi itu sebagai kunci utama tabel — tetapi hanya jika nilai di kolom ini akan selalu berbeda untuk setiap catatan. Anda tidak dapat memiliki nilai duplikat dalam kunci utama. Misalnya, jangan gunakan nama orang sebagai kunci utama, karena nama tidak unik. Anda dapat dengan mudah memiliki dua orang dengan nama yang sama dalam tabel yang sama.

Kunci utama harus selalu memiliki nilai. Jika nilai kolom dapat menjadi tidak ditetapkan atau tidak diketahui (nilai yang hilang) di beberapa titik, nilai tersebut tidak dapat digunakan sebagai komponen dalam kunci utama.

Anda harus selalu memilih kunci utama yang nilainya tidak akan berubah. Dalam database yang menggunakan lebih dari satu tabel, kunci utama tabel dapat digunakan sebagai referensi dalam tabel lain. Jika kunci utama berubah, perubahan juga harus diterapkan di mana saja kunci dirujuk. Menggunakan kunci utama yang tidak akan berubah mengurangi kemungkinan kunci utama mungkin menjadi tidak sinkron dengan tabel lain yang mereferensikannya.

Seringkali, angka unik semena-mena digunakan sebagai kunci utama. Misalnya, Anda dapat menetapkan setiap pesanan nomor pesanan yang unik. Satu-satunya tujuan nomor pesanan adalah untuk mengidentifikasi pesanan. Setelah ditetapkan, tidak pernah berubah.

Jika Anda tidak keberatan dengan kolom atau sekumpulan kolom yang mungkin menjadi kunci utama yang baik, pertimbangkan untuk menggunakan kolom yang memiliki tipe data AutoNumber. Saat Anda menggunakan tipe data AutoNumber, Access secara otomatis menetapkan nilai untuk Anda. Pengidentifikasi semacam itu tidak ada faktanya; tidak berisi informasi faktual yang menjelaskan baris yang diwakilinya. Pengidentifikasi tanpa fakta ideal untuk digunakan sebagai kunci utama karena pengidentifikasi tidak berubah. Kunci utama yang berisi fakta tentang baris — nomor telepon atau nama pelanggan, misalnya — lebih mungkin berubah, karena informasi faktual itu sendiri mungkin berubah.



Gambar 4. 5 Contoh Tabel Kunci Utama

1. Kolom yang diatur ke tipe data AutoNumber sering membuat kunci utama yang baik. Tidak ada dua ID produk yang sama.

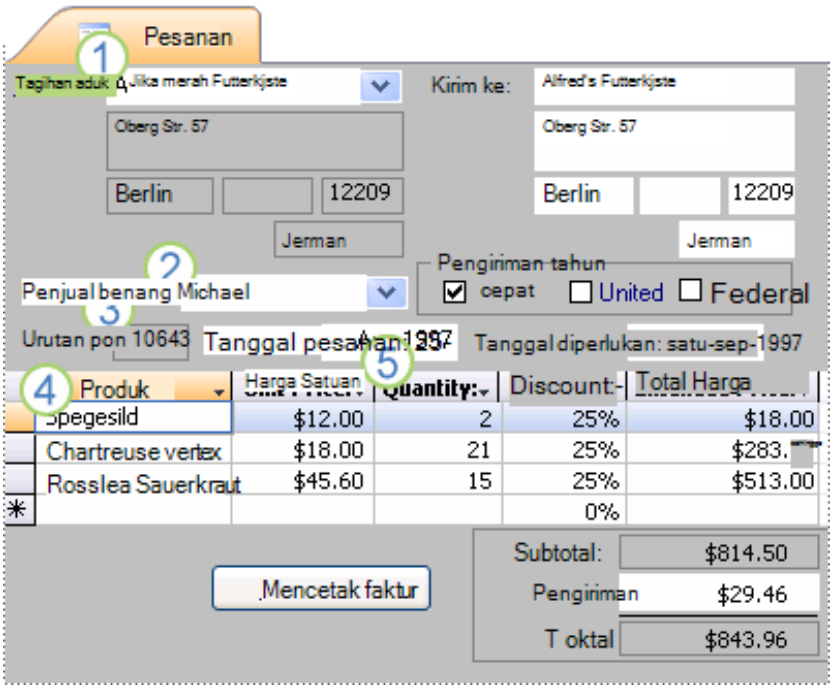
Dalam beberapa kasus, Anda mungkin ingin menggunakan dua bidang atau lebih yang, bersama-sama, menyediakan kunci utama tabel. Misalnya, tabel Detail Pesanan yang menyimpan item baris untuk pesanan akan menggunakan dua kolom dalam kunci utamanya: ID Pesanan dan ID Produk. Ketika kunci utama menggunakan lebih dari satu kolom, kunci tersebut juga disebut kunci komposit.

Untuk database penjualan produk, Anda bisa membuat kolom AutoNumber untuk setiap tabel agar berfungsi sebagai kunci utama: IDProduk untuk tabel Produk, IDPesanan untuk tabel Pesanan, ID Pelanggan untuk tabel Pelanggan, dan ID Pemasok untuk tabel Pemasok.

Gambar 4. 6 Contoh Hasil Dari Kunci Utama

Membuat hubungan tabel

Setelah membagi informasi menjadi tabel, Anda memerlukan cara untuk menyampulkan informasi kembali dengan cara yang bermakna. Misalnya, formulir berikut ini menyertakan informasi dari beberapa tabel.



Gambar 4. 7 Contoh Menghubungkan Tabel

1. Informasi di formulir ini berasal dari tabel Pelanggan...

2. ... tabel Karyawan...

3. ... tabel Pesanan...

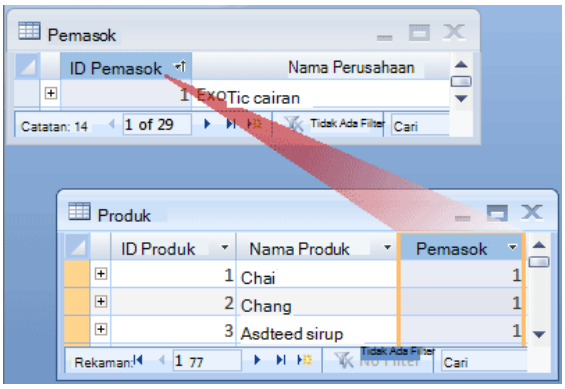
4. ... tabel Produk...

5. ... dan tabel Detail Pesanan.

Access adalah sistem manajemen database relasionais. Dalam database relasialis, Anda membagi informasi menjadi tabel berbasis subjek yang terpisah. Lalu, gunakan hubungan tabel untuk menyampulkan informasi sesuai kebutuhan.

Membuat hubungan satu ke banyak

Pertimbangkan contoh ini: tabel Pemasok dan Produk dalam database pesanan produk. Pemasok dapat menyediakan sejumlah produk. Hal ini mengikuti bahwa untuk setiap pemasok yang diwakili dalam tabel Pemasok, dapat ada banyak produk yang diwakili dalam tabel Produk. Oleh karena itu, hubungan antara tabel Pemasok dan tabel Produk adalah hubungan satu ke banyak.



Gambar 4. 8 Menghubungkan Satu ke Banyak

Untuk mewakili hubungan satu ke banyak dalam desain database Anda, ambil kunci utama di sisi "satu" hubungan dan tambahkan sebagai kolom atau kolom tambahan ke tabel di sisi "banyak" hubungan. Dalam hal ini, misalnya, Anda menambahkan kolom ID Pemasok dari tabel Pemasok ke tabel Produk. Access kemudian dapat menggunakan nomor ID pemasok dalam tabel Produk untuk menemukan pemasok yang tepat untuk setiap produk.

Kolom ID Pemasok dalam tabel Produk disebut kunci asing. Kunci asing adalah kunci utama tabel lain. Kolom ID Pemasok dalam tabel Produk adalah kunci asing karena juga merupakan kunci utama dalam tabel Pemasok.

Gambar 4. 9 Contoh Hasil Dari Satu Menjadi Banyak

Anda menyediakan dasar untuk menggabungkan tabel terkait dengan menetapkan pemasangan kunci utama dan kunci asing. Jika Anda tidak yakin tabel mana yang harus berbagi kolom umum, mengidentifikasi hubungan satu ke banyak memastikan bahwa dua tabel yang terlibat akan memerlukan kolom bersama.

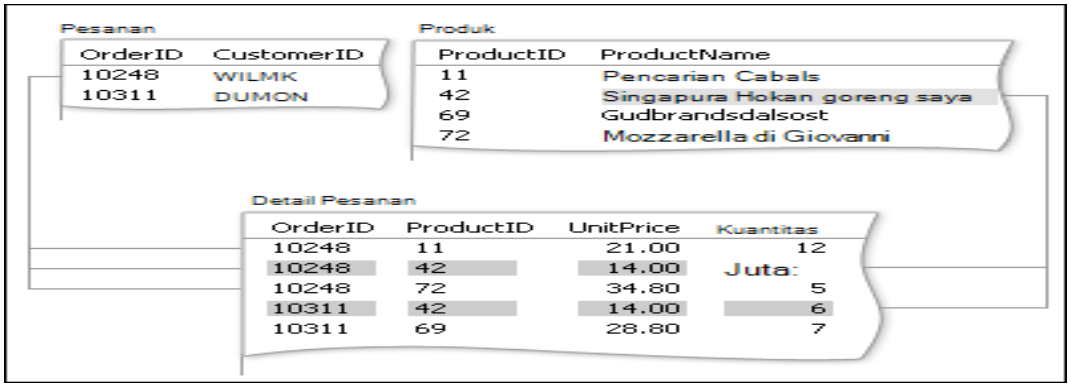
Membuat hubungan banyak ke banyak

Pertimbangkan hubungan antara tabel Produk dan tabel Pesanan.

Sebuah pesanan bisa berisi lebih dari satu produk. Di sisi lain, sebuah produk bisa muncul di banyak pesanan. Karena itu, untuk tiap rekaman dalam tabel Pesanan, bisa terdapat banyak rekaman dalam tabel Produk. Dan untuk setiap catatan dalam tabel Produk, bisa ada banyak rekaman dalam tabel Pesanan. Tipe hubungan ini disebut hubungan banyak ke banyak karena untuk produk apa pun, bisa ada banyak pesanan; dan untuk pesanan apa pun, bisa ada banyak produk. Perhatikan bahwa untuk mendeteksi hubungan banyak ke banyak di antara tabel Anda, penting bagi Anda untuk mempertimbangkan kedua sisi hubungan.

Subjek dari dua tabel — pesanan dan produk — memiliki hubungan banyak ke banyak. Hal ini menyuguhkan masalah. Untuk memahami masalah, bayangkan apa yang akan terjadi jika Anda mencoba membuat hubungan antara kedua tabel dengan menambahkan bidang ID Produk ke tabel Pesanan. Untuk memiliki lebih dari satu produk per pesanan, Anda memerlukan lebih dari satu catatan dalam tabel Pesanan per pesanan. Anda akan mengulangi informasi pesanan untuk setiap baris yang berkaitan dengan satu urutan — menghasilkan desain yang tidak efisien yang dapat mengakibatkan data yang tidak akurat. Anda mengalami masalah yang sama jika Anda meletakkan bidang ID Pesanan dalam tabel Produk — Anda akan memiliki lebih dari satu catatan dalam tabel Produk untuk setiap produk. Bagaimana Anda mengatasi masalah ini?

Jawabannya adalah dengan membuat tabel ketiga, sering disebut tabel persimpangan, yang memecah hubungan banyak ke banyak menjadi dua hubungan satu ke banyak. Anda menyisipkan kunci utama dari masing-masing dua tabel ke tabel ketiga tersebut. Akibatnya, tabel ketiga merekam setiap kemunculan atau contoh hubungan.



Gambar 4. 10 Contoh Menghubungkan Banyak ke Banyak

Setiap rekaman dalam tabel Detail Pesanan mewakili satu item baris dalam urutan. Kunci utama tabel Detail Pesanan terdiri dari dua bidang — kunci asing dari tabel Pesanan dan Produk. Menggunakan bidang ID Pesanan saja tidak berfungsi sebagai kunci utama untuk tabel ini, karena satu pesanan bisa memiliki banyak item baris. ID Pesanan diulangi untuk setiap item baris dalam urutan, sehingga bidang tidak berisi nilai unik. Menggunakan bidang ID Produk saja tidak berfungsi, karena satu produk dapat muncul pada banyak pesanan yang berbeda. Namun bersama-sama, dua bidang tersebut selalu menghasilkan nilai yang unik untuk setiap catatan.

Dalam database penjualan produk, tabel Pesanan dan tabel Produk tidak terkait satu sama lain secara langsung. Sebaliknya, data terkait secara tidak langsung melalui tabel Detail Pesanan. Hubungan banyak ke banyak antara pesanan dan produk dinyatakan dalam database dengan menggunakan dua hubungan satu ke banyak:

- Tabel Pesanan dan tabel Detail Pesanan memiliki hubungan satu ke banyak. Setiap pesanan dapat memiliki lebih dari satu item baris, tetapi setiap item baris hanya tersambung ke satu pesanan.
- Tabel Produk dan tabel Detail Pesanan memiliki hubungan satu ke banyak. Setiap produk dapat memiliki banyak item baris yang terkait dengannya, tetapi setiap item baris hanya merujuk ke satu produk.

Dari tabel Detail Pesanan, Anda dapat menentukan semua produk pada pesanan tertentu. Anda juga dapat menentukan semua pesanan untuk produk tertentu.

Setelah menggabungkan tabel Detail Pesanan, daftar tabel dan bidang mungkin terlihat seperti ini:

Gambar 4. 11 Contoh Hasil Dari Menghubungan Banyak ke Banyak

Membuat hubungan satu ke satu

Tipe hubungan lainnya adalah hubungan satu ke satu. Misalnya, Anda perlu merekam beberapa informasi produk tambahan khusus yang jarang Anda perlukan atau yang hanya berlaku untuk beberapa produk. Karena Anda tidak sering membutuhkan informasi, dan karena menyimpan informasi dalam tabel Produk akan mengakibatkan ruang kosong untuk setiap produk yang tidak diterapkannya, Anda menempatkannya di tabel terpisah. Seperti tabel Produk, Anda menggunakan IDProduk sebagai kunci utama. Hubungan antara tabel tambahan ini dan tabel Produk adalah hubungan satu ke satu. Untuk setiap catatan dalam tabel Produk, ada satu rekaman yang cocok dalam tabel tambahan. Saat Anda memang mengidentifikasi hubungan tersebut, kedua tabel harus berbagi bidang yang sama.

Saat Anda mendeteksi perlunya hubungan satu ke satu dalam database Anda, pertimbangkan apakah Anda bisa meletakkan informasi dari dua tabel bersama-sama dalam satu tabel. Jika Anda tidak ingin melakukan itu karena beberapa alasan, mungkin karena itu akan menghasilkan banyak ruang kosong, daftar berikut ini memperlihatkan bagaimana Anda akan mewakili hubungan dalam desain Anda:

- Jika kedua tabel memiliki subjek yang sama, Anda mungkin bisa menyetel hubungan dengan menggunakan kunci utama yang sama di kedua tabel.
- Jika kedua tabel memiliki subjek yang berbeda dengan kunci utama yang berbeda, pilih salah satu tabel (salah satu tabel) dan sisipkan kunci utamanya di tabel lain sebagai kunci asing.

Menentukan hubungan antar tabel membantu Anda memastikan bahwa Anda memiliki tabel dan kolom yang tepat. Ketika hubungan satu ke satu atau satu ke banyak ada, tabel yang dilibatkan perlu berbagi kolom atau kolom umum. Ketika hubungan banyak ke banyak ada, tabel ketiga diperlukan untuk mewakili hubungan.

Menyempurnakan desain

Setelah memiliki tabel, bidang, dan hubungan yang diperlukan, Anda harus membuat dan mengisi tabel dengan data sampel dan mencoba bekerja dengan informasi: membuat kueri, menambahkan catatan baru, dan seterusnya. Melakukan ini membantu menyoroti masalah potensial — misalnya, Anda mungkin perlu menambahkan kolom yang lupa disisipkan selama fase desain, atau Mungkin ada tabel yang harus Anda pisahkan menjadi dua tabel untuk menghapus duplikasi.

Lihat apakah Anda bisa menggunakan database untuk mendapatkan jawaban yang Anda inginkan. Buat draf kasar formulir dan laporan Anda dan lihat apakah formulir dan laporan memperlihatkan data yang Anda harapkan. Cari duplikasi data yang tidak diperlukan dan, ketika Anda menemukannya, ubah desain Anda untuk menghilangkannya.

Saat mencoba database awal, Anda mungkin akan menemukan ruang untuk penyempurnaan. Berikut adalah beberapa hal untuk diperiksa:

- Apakah Anda lupa kolom apa pun? Jika demikian, apakah informasi tersebut termasuk dalam tabel yang sudah ada? Jika ini adalah informasi tentang hal lain, Anda mungkin perlu membuat tabel lain. Buat kolom untuk setiap item informasi yang perlu Anda lacak. Jika informasi tidak dapat dihitung dari kolom lain, kemungkinan Anda akan memerlukan kolom baru untuk kolom tersebut.
- Apakah kolom apa pun tidak diperlukan karena kolom dapat dihitung dari bidang yang sudah ada? Jika item informasi dapat dihitung dari kolom lain yang sudah ada — harga diskon yang dihitung dari harga eceran, misalnya — biasanya lebih baik untuk melakukan hal itu, dan menghindari pembuatan kolom baru.
- Apakah Anda berulang kali memasukkan informasi duplikat di salah satu tabel? Jika demikian, Anda mungkin perlu membagi tabel menjadi dua tabel yang memiliki hubungan satu ke banyak.
- Apakah Anda memiliki tabel dengan banyak bidang, jumlah rekaman terbatas, dan banyak bidang kosong dalam catatan individual? Jika demikian, pikirkan tentang mendesain ulang tabel sehingga memiliki lebih sedikit bidang dan rekaman lainnya.
- Apakah setiap item informasi telah dipecah menjadi bagian terkecil yang berguna? Jika Anda perlu melaporkan, mengurutkan, mencari, atau menghitung item informasi, letakkan item tersebut di kolomnya sendiri.
- Apakah setiap kolom berisi fakta tentang subjek tabel? Jika kolom tidak berisi informasi tentang subjek tabel, kolom tersebut berada dalam tabel berbeda.
- Apakah semua hubungan antar tabel diwakili, baik menurut bidang umum atau menurut tabel ketiga? Hubungan satu ke satu dan satu ke-banyak memerlukan kolom umum. Hubungan banyak ke banyak memerlukan tabel ketiga.

Menyempurnakan tabel Produk

Anggaplah bahwa setiap produk dalam database penjualan produk termasuk dalam kategori umum, seperti minuman, bumbu, atau makanan laut. Tabel Produk dapat menyertakan bidang yang memperlihatkan kategori setiap produk.

Misalnya setelah memeriksa dan menyempurnakan desain database, Anda memutuskan untuk menyimpan deskripsi kategori beserta namanya. Jika Anda menambahkan bidang Deskripsi Kategori ke tabel Produk, Anda harus mengulangi setiap deskripsi kategori untuk setiap produk yang termasuk dalam kategori — ini bukan solusi yang baik.

Solusi yang lebih baik adalah membuat Kategori menjadi subjek baru bagi database untuk dilacak, dengan tabelnya sendiri dan kunci utamanya sendiri. Anda kemudian dapat menambahkan kunci utama dari tabel Kategori ke tabel Produk sebagai kunci asing.

Tabel Kategori dan Produk memiliki hubungan satu ke banyak: kategori dapat menyertakan lebih dari satu produk, tetapi produk hanya dapat termasuk dalam satu kategori.

Saat Anda meninjau struktur tabel Anda, waspadai grup berulang. Misalnya, pertimbangkan tabel yang berisi kolom berikut:

- Product ID
- Nama
- ID Produk1
- Name1
- ID Produk2
- Nama2
- ID Produk3
- Name3

Di sini, setiap produk adalah grup kolom berulang yang berbeda dari yang lain hanya dengan menambahkan angka ke akhir nama kolom. Ketika melihat kolom bernomor seperti ini, Anda harus mengunjungi kembali desain Anda.

Desain seperti itu memiliki beberapa kelemahan. Sebagai permulaan, ini memaksa Anda untuk menempatkan batas atas pada jumlah produk. Segera setelah melampaui batas tersebut, Anda harus menambahkan sekelompok kolom baru ke struktur tabel, yang merupakan tugas administratif utama.

Masalah lain adalah bahwa pemasok yang memiliki lebih sedikit dari jumlah maksimum produk akan membuang beberapa ruang, karena kolom tambahan akan kosong. Kelemahan yang paling serius dengan desain seperti itu adalah membuat banyak tugas sulit dilakukan, seperti mengurutkan atau mengindeks tabel berdasarkan ID produk atau nama.

Setiap kali Anda melihat grup berulang meninjau desain dengan cermat dengan mata memisahkan tabel menjadi dua. Dalam contoh di atas lebih baik menggunakan dua tabel, satu untuk pemasok dan satu untuk produk, ditautkan oleh ID pemasok.

Menerapkan aturan normalisasi

Anda bisa menerapkan aturan normalisasi data (terkadang hanya disebut aturan normalisasi) sebagai langkah berikutnya dalam desain Anda. Gunakan aturan ini untuk melihat apakah tabel Anda disusun dengan benar. Proses penerapan aturan ke desain database Anda disebut menormalkan database, atau hanya normalisasi.

Normalisasi paling berguna setelah Anda mewakili semua item informasi dan telah sampai pada desain pendahuluan. Idenya adalah untuk membantu Anda memastikan bahwa Anda telah membagi item informasi Anda ke dalam tabel yang sesuai. Normalisasi apa yang tidak bisa dilakukan adalah memastikan bahwa Anda memiliki semua item data yang benar untuk dimulai.

Anda menerapkan aturan secara berurut, pada setiap langkah memastikan bahwa desain Anda tiba di salah satu dari apa yang dikenal sebagai "formulir normal." Lima bentuk normal diterima secara luas — bentuk normal pertama melalui bentuk normal kelima. Artikel ini diperluas di tiga pertama, karena semuanya diperlukan untuk sebagian besar desain database.

Formulir normal pertama

Formulir normal pertama menyatakan bahwa pada setiap irisan baris dan kolom dalam tabel di sana, ada nilai tunggal, dan tidak pernah ada daftar nilai. Misalnya, Anda tidak dapat memiliki bidang bernama Harga di mana Anda menempatkan lebih dari satu Harga. Jika Anda menganggap setiap irisan baris dan kolom sebagai sel, setiap sel hanya bisa menampung satu nilai.

Formulir normal kedua

Formulir normal kedua mengharuskan setiap kolom non-kunci sepenuhnya bergantung pada seluruh kunci utama, bukan hanya sebagian kunci. Aturan ini berlaku ketika Anda memiliki kunci utama yang terdiri dari lebih dari satu kolom. Misalnya, anggaplah Anda memiliki tabel yang berisi kolom berikut ini, di mana ID Pesanan dan ID Produk membentuk kunci utama:

- ID Pesanan (kunci utama)
- ID produk (kunci utama)
- Nama Produk

Desain ini melanggar formulir normal kedua, karena Nama Produk bergantung pada ID Produk, tetapi tidak pada ID Pesanan, sehingga tidak bergantung pada seluruh kunci utama. Anda harus menghapus Nama Produk dari tabel. Ini termasuk dalam tabel yang berbeda (Produk).

Formulir normal ketiga

Formulir normal ketiga mengharuskan agar tidak hanya setiap kolom non-kunci yang bergantung pada seluruh kunci utama, tetapi kolom non-kunci tidak terpisah satu sama lain.

Cara lain untuk mengatakan ini adalah bahwa setiap kolom non-kunci harus bergantung pada kunci utama dan tidak ada selain kunci utama. Misalnya, anggaplah Anda memiliki tabel yang berisi kolom berikut ini:

- PRODUCTID (kunci utama)
- Nama
- SRP
- Discount

Asumsikan bahwa Diskon tergantung pada harga eceran yang disarankan (SRP). Tabel ini melanggar formulir normal ketiga karena kolom non-kunci, Diskon, bergantung pada kolom non-kunci lainnya, SRP. Independensi kolom berarti Anda harus bisa mengubah kolom non-kunci apa pun tanpa memengaruhi kolom lain. Jika Anda mengubah nilai dalam bidang SRP, Diskon akan berubah sesuai keinginan, sehingga melanggar aturan tersebut. Dalam hal ini Diskon harus dipindahkan ke tabel lain yang ditujukan pada SRP.

4.2 Konsep Normalisasi

Normalisasi merupakan sebuah teknik logical desain dalam sebuah basis data yang mengelompokkan atribut dari berbagai entitas dalam suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi/pengulangan data) serta sebagian besar ambiguity bisa dihilangkan.

Atau pengertian singkatny, Normalisasi Databse adalah proses pengelompokan atribut data yang membentuk entitas sederhana, nonredundan, fleksibel, dan mudah beradaptasi, Sehingga dapat dipastikan bahwa database yang dibuat berkualitas baik.

Normalisasi database terdiri dari banyak bentuk, dalam ilmu basis data ada setidaknya 9 bentuk normalisasi yang ada yaitu 1NF, 2NF, 3NF, EKNF, BCNF, 4NF, 5NF, DKNF, dan 6NF.

Database 1NF, 2NF, dan 3NF akan sering ditemui ketika akan membuat sebuah database yang optimal. Jika Anda ingin menjadi seorang Database Administrator (DBA), harus tahu bagaimana cara normalisasi database yang optimal. Misalkan suatu saat ketika website yang Anda buat mengalami penurunan kinerja, mungkin Anda akan ditanya apakah database tersebut sudah dinormalisasi dengan benar.

Tujuan Normalisasi Database

Tujuannya adalah untuk menghilangkan dan mengurangi redudansi data dan tujuan yang kedua adalah memastikan dependensi data (Data berada pada tabel yang tepat). Jika data dalam database tersebut belum di normalisasi maka akan terjadi 3 kemungkinan yang akan merugikan sistem secara keseluruhan.

1. INSERT Anomali : Situasi dimana tidak memungkinkan memasukkan beberapa jenis data secara langsung di database.
2. DELETE Anomali: Penghapusan data yang tidak sesuai dengan yang diharapkan, artinya data yang harusnya tidak terhapus mungkin ikut terhapus.
3. UPDATE Anomali: Situasi dimana nilai yang diubah menyebabkan inkonsistensi database, dalam artian data yang diubah tidak sesuai dengan yang diperintahkan atau yang diinginkan.

Database Seperti Apa yang dinormalisasi

Tidak semua database bisa dinormalisasi, hanya tipe “relational database“ yang bisa dinormalisasi. Banyak vendor DBMS (Database Management System) diantaranya Oracle, MySQL, SQL Server, PostgreSQL, dll.

Tahapan Normalisasi Database

Untuk melakukan normalisasi database kita harus mengidentifikasi data seperti apa yang akan disimpan, dan berikut adalah contohnya:

1. Bentuk Tidak Normal (unnormalize)

   Bentuk tidak normal (unnormalized) merupakan kumpulan data yang direkam tidak ada keharusan dengan mengikuti suatu format tertentu.

   Pada bentuk tidak normal terdapat repeating group (Pengulangan Group), sehingga pada kondisi ini data menjadi permasalahan dalam melakukan manipulasi data (insert, update, dan delete) atau biasa disebut anomali.

id_member	nama_member	tipe_member	keterangan_member	id_theater	nama_theater	kota	id_movie	nama_movie	id_faktur	waktu_tayang	harga_tiket	qty_tiket
111	Muri	EPC	Epic	T01	Paris Van Java	Bandung	M01	Orang kaya baru	F001	22/01/2019 17:00	35.000	3
							M03	Twice Land	F002	22/01/2019 19:00	35.000	3
114	Luga	ELT	Elite	T02	Grand Indonesia	Jakarta	M03	Twice Land	F003	22/01/2019 19:00	35.000	2
							M04	Escape Room	F004	22/01/2019 21:00	35.000	2
113	Rigo	MTC	Mythic	T01	Paris Van Java	Bandung	M01	Orang kaya baru	F005	22/01/2019 17:00	35.000	3
							M02	Terlalu Tampan	F006	22/01/2019 19:00	35.000	3
							M05	Welcome Home	F007	22/01/2019 21:00	35.000	3
114	Muri	EPC	Epic	T02	Grand Indonesia	Jakarta	M02	Terlalu Tampan	F008	22/01/2019 19:00	35.000	2

Gambar 4. 12 Contoh Bentuk Tidak Normal

2. 1NF / First Normal Form

1NF mensyaratkan beberapa kondisi dalam sebuah database, berikut adalah fungsi dari bentuk normal pertama ini.

- Menghilangkan duplikasi kolom dari tabel yang sama.
- Buat tabel terpisah untuk masing-masing kelompok data terkait dan mengidentifikasi setiap baris dengan kolom yang unik (primary key).

id_member	nama_member	tipe_member	keterangan_member	id_theater	nama_theater	kota	id_movie	nama_movie	id_faktur	waktu_tayang	harga_tiket	qty_tiket
111	Muri	EPC	Epic	T01	Paris Van Java	Bandung	M01	Orang kaya baru	F001	22/01/2019 17:00	35.000	3
112	Muri	EPC	Epic	T02	Paris Van Java	Bandung	M03	Twice Land	F002	22/01/2019 19:00	35.000	3
114	Luga	ELT	Elite	T02	Grand Indonesia	Jakarta	M03	Twice Land	F003	22/01/2019 19:00	35.000	2
115	Luga	ELT	Elite	T03	Grand Indonesia	Jakarta	M04	Escape Room	F004	22/01/2019 21:00	35.000	2
113	Rigo	MTC	Mythic	T01	Paris Van Java	Bandung	M01	Orang kaya baru	F005	22/01/2019 17:00	35.000	3
114	Rigo	MTC	Mythic	T02	Paris Van Java	Bandung	M02	Terlalu Tampan	F006	22/01/2019 19:00	35.000	3
115	Rigo	MTC	Mythic	T03	Paris Van Java	Bandung	M05	Welcome Home	F007	22/01/2019 21:00	35.000	3
114	Muri	EPC	Epic	T02	Grand Indonesia	Jakarta	M02	Terlalu Tampan	F008	22/01/2019 19:00	35.000	2

Gambar 4. 13 Contoh 1NF / First Normal Form

Pada intinya bentuk normalisasi 1NF ini mengelompokkan beberapa tipe data atau kelompok data yang sejenis agar dapat dipisahkan sehingga anomali data dapat di atasi.

Contoh adalah ketika kita ingin menghapus, mengupdate, atau menambahkan data peminjam, maka kita tidak bersinggungan dengan data buku atau data penerbit. Sehingga inkonsistensi data dapat mulai di jaga.

3. 2NF

Syarat untuk menerapkan normalisasi bentuk kedua ini adalah data telah dibentuk dalam 1NF, berikut adalah beberapa fungsi normalisasi 2NF.

- Menghapus beberapa subset data yang ada pada tabel dan menempatkan mereka pada tabel terpisah.
- Menciptakan hubungan antara tabel baru dan tabel lama dengan menciptakan foreign key.
- Tidak ada atribut dalam tabel yang secara fungsional bergantung pada candidate key tabel tersebut.

No Polisi	Warna	Merek	Tahun
B3117LB	Biru	Supra X	2005
B2121AA	Merah	Supra X	2005

Mekanik ID	Nama Mekanik
DDE	Djoko Dewanto

Kode Parts	Nama Parts	Harga
20W501000CC	Oli Top 1 000cc	27000
SERV001	Engine Tune Up	25000

No Faktur	Tanggal	No Polisi	Mekanik ID	Kode Parts	Kuantum	Harga	Discount	Potongan
05103214	25/10/2005	B3117LB	DDE	20W501000CC	2	27000	1000	2000
05103214	25/10/2005	B3117LB	DDE	SERV001	1	25000	2000	2000
05103215	25/10/2005	B2121AA	DDE	SERV001	1	25000	2000	0

Gambar 4. 14 Contoh 2NF

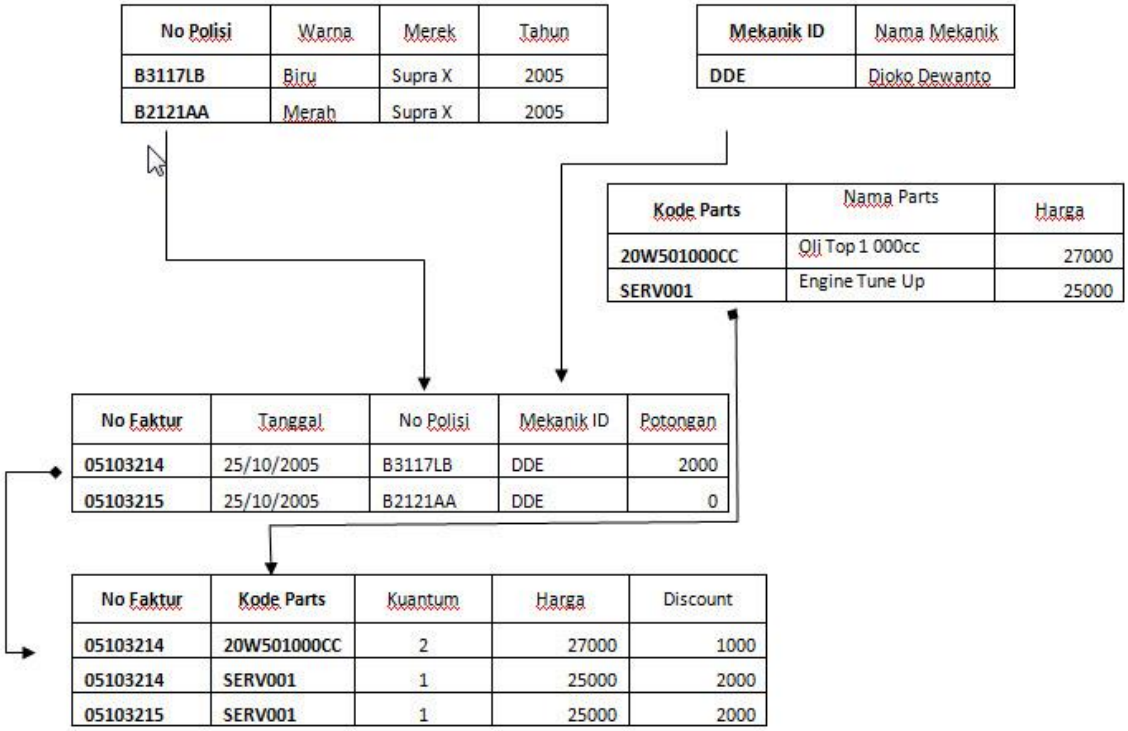
Bentuk normal kedua dengan melakukan dekomposisi tabel diatas menjadi beberapa tabel dan mencari kunci primer dari tiap-tiap tabel tersebut dan atribut kunci haruslah unik.

4. 3NF

Pada 3NF tidak diperkenankan adanya partial "transitive dependency" dalam sebuah tabel. Transitive dependency biasanya terjadi pada tabel hasil relasi, atau kondisi dimana terdapat tiga atribut A, B, C. Kondisinya adalah A ⇒ B dan B ⇒ C. Maka C dikatakan sebagai transitive dependency terhadap A melalui B.

Intinya pada 3NF ini, jika terdapat suatu atribut yang tidak bergantung pada primary key tapi bergantung pada field yang lain maka atribut-atribut tersebut perlu dipisah ke tabel baru.

Contohnya ada pada atribut qty, kolom tersebut tidak bergantung langsung pada primary key kode_faktur melainkan bergantung pada kolom kode_barang. Jadi setelah dinormalisasi 3NF akan menghasilkan tabel berikut:



No Polisi	Warna	Merek	Tahun
B3117LB	Biru	Supra X	2005
B2121AA	Merah	Supra X	2005

Mekanik ID	Nama Mekanik
DDE	Dioko Dewanto

Kode Parts	Nama Parts	Harga
20W501000CC	Oli Top 1 000cc	27000
SERV001	Engine Tune Up	25000

No Faktur	Tanggal	No Polisi	Mekanik ID	Potongan
05103214	25/10/2005	B3117LB	DDE	2000
05103215	25/10/2005	B2121AA	DDE	0

No Faktur	Kode Parts	Kuantum	Harga	Discount
05103214	20W501000CC	2	27000	1000
05103214	SERV001	1	25000	2000
05103215	SERV001	1	25000	2000

Gambar 4. 15 Contoh 3NF

Dari gambar tabel di atas dapat dilihat pada tahap normalisasi 3NF menghasilkan 1 tabel baru dari hasil pemecahan tabel transaksi yaitu tabel detail barang yang isinya menampung barang-barang yang dibeli.

Kolom harga pada tabel detail barang digunakan untuk menyimpan harga barang pada saat proses transaksi. Jadi, meskipun kolom harga pada tabel barang berubah (naik/turun), harga barang yang ada pada tabel detail barang tidak ikut berubah (fixed). Bayangkan jika kita tidak menambahkan kolom harga pada pada tabel detail barang, maka yang terjadi total invoice dari transaksi akan berubah seiring berubahnya harga barang.

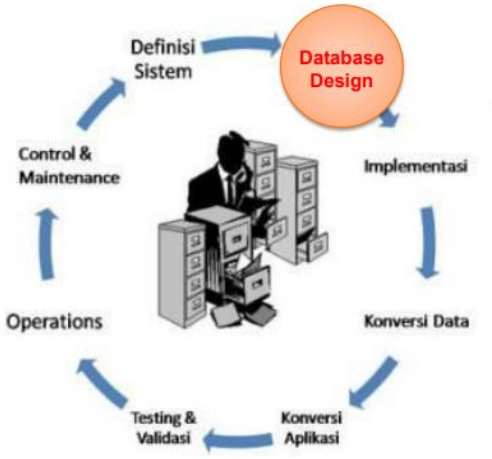
5. BCNF Boyce–Codd normal form

Merupakan sebuah teknik normalisasi database yang sering disebut 3.5NF, memiliki hubungan yang sangat erat dengan bentuk 3NF. Pada dasarnya adalah untuk menghandle anomali dan overlooping yang tidak dapat di handle dalam bentuk 3NF. Normalisasi database bentuk ini tergantung dari kasus yang disediakan, tidak semua tabel wajib di normalisasi dalam bentuk BCNF.

Untuk tabel untuk memenuhi Bentuk Normal Boyce-Codd, harus memenuhi dua kondisi berikut:

- yaitu Table harus dalam Bentuk Normal Ketiga.
- Dan, untuk ketergantungan apa pun A → B, A harus menjadi super key. Poin kedua terdengar agak rumit, kan? Dengan kata sederhana, itu berarti, bahwa untuk ketergantungan A → B, A tidak dapat menjadi atribut non-prima, jika B adalah atribut utama.

4.3 Metode Desain Basis Data



Gambar 4. 16 Contoh Daur Hidup Basis Data

- System Definition : Mendefinisikan Scope dari sistem basis data, pemakai dan aplikasi Antarmuka untuk pemakai, batasan response time, kebutuhan penyimpan dan pemrosesan diidentifikasi.

- Database Design Pada akhir dari tahap ini , desain konseptual, desain logika dan fisik dari sistem basis data dari DBMS sudah siap.

- Database Implementation Meliputi proses menentukan definisi basis data eksternal, konseptual dan internal, membuat file basis data kosong dan implementasi aplikasi perangkat lunak.

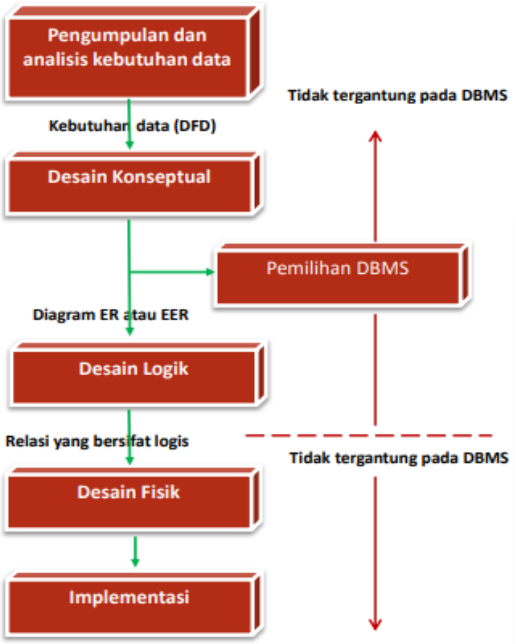
- Loading or Data Conversion Basis data dipopulasikan dengan menyimpan data langsung atau mengubah file yang sudah ada ke format sistem basis data.

- Application Conversion Aplikasi perangkat lunak dari sistem lama dikonversikan ke sistem baru. ′ Testing and Validation Sistem baru diuji coba dan divalidasi

- Operation Sistem basis data dan aplikasi dioperasikan. Biasanya sistem lama dan baru dioperasikan secara paralel dalam beberapa waktu.

- Monitoring and Maintenance Selama tahap operasional, sistem secara tetap dimonitor dan dipelihara. Perubahan dan pengembangan dapat terjadi baik pada isi data maupun aplikasi perangkat lunak

6 Tahap Proses Desain Database



Gambar 4. 17 Contoh 6 Tahap Proses Desain Database

TAHAP 1:

Pengumpulan Dan Analisis Kebutuhan Data Tools : HIPO, SADT, DFD, FLOWDOCUMENT, (HIERARCHICAL) Aktifitas yang dilakukan :

- Area aplikasi mayor dan kelompok pemakai yang akan menggunakan basis data atau pekerjaan / aplikasinya.

- Dokumen yang sudah ada yang berhubungan dengan aplikasi dipelajari dan dianalisa. Dokumen lain seperti police manual, form, report dan struktur organisasi ditinjau kembali untuk menentukan dan menguji apakah dokumen-dokumen tersebut berpengaruh terhadap kumpulan data dan proses spesifikasi.

- Lingkungan operasi saat ini dan rencana penggunaan informasi. Menganalisa tipe transaksi dan frekuensi penggunaannya dan aliran informasi dalam sistem. Karakteristik geografi seperti pemakai, transaksi asli, tujuan pelaporan. Data input dan output diperinci.

- Penulisan respon dari kuesioner pemakai potensial untuk mendapatkan informasi yg berharga

TAHAP 2:

Desain Database Konseptual Aktifitas Paralel

1. Desain Skema Konseptual

Memberikan gambaran yang lengkap dari struktur basis data yaitu arti, hubungan, dan batasan-batasan.

- Conceptual schema bersifat tetap.
- Alat komunikasi antar pemakai basis data, designer, dan analis.
- Harus bersifat: Mampu menyatakan relationship, batasan-batasan, Diagram, Formal, minimum dalam menyatakan spesifikasi data (tidak ada duplikasi), Simple
-

Karakteristik :

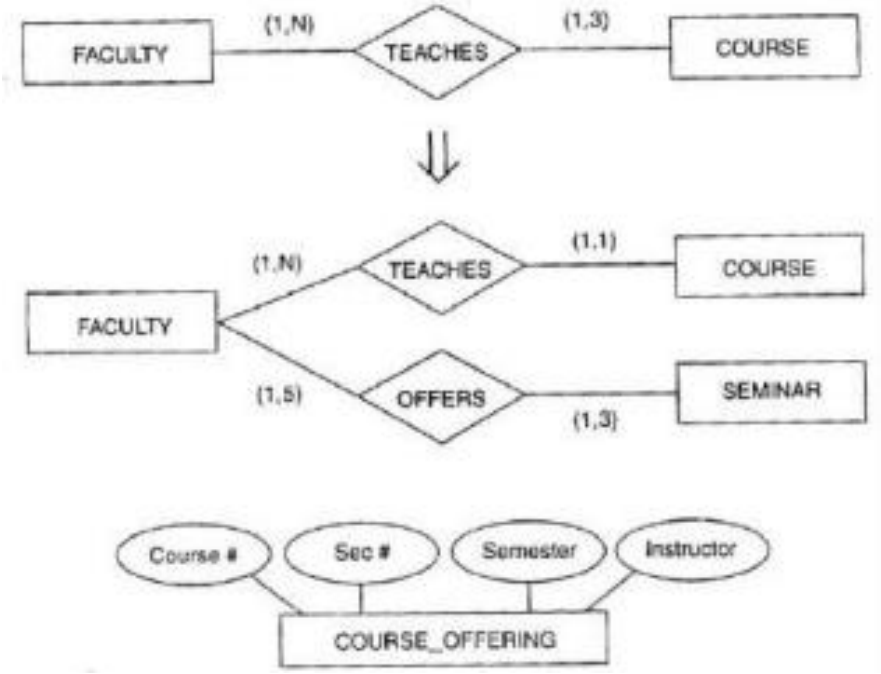
- Expressiveness : Model data cukup ekspresif untuk membedakan perbedaan data,relationship dan konstrain.
- Simplicity & Understandability Model cukup sederhana untuk pemakai yang kurang mengerti.
- Minimality Model mempunyai sejumlah kecil konsep dasar yang berbeda dan tidak overlapping.
- Diagrammatic Representation Menggunakan notasi diagram untuk menampilkan skema konseptual yang mudah di interpretasikan.
- Formality Model data harus merepresentasikan formal data

Strategi :

- Top Down: Mulai dengan beberapa high level entity type, Bagi lagi (top down) menjadi beberapa lower-level entity type dan Relationship type.
- Bottom Up: Mulai dengan atribut, Kelompokkan menjadi entity type & relationship type, Tambahkan relationship-relationship.
- Inside Out: Bentuk khusus dari bottom-up, Mula-mula ditentukan entity type yang merupakan pusat/bagian terpenting tambahkan entity type dan relationship lain yang berhubungan satu sama lain

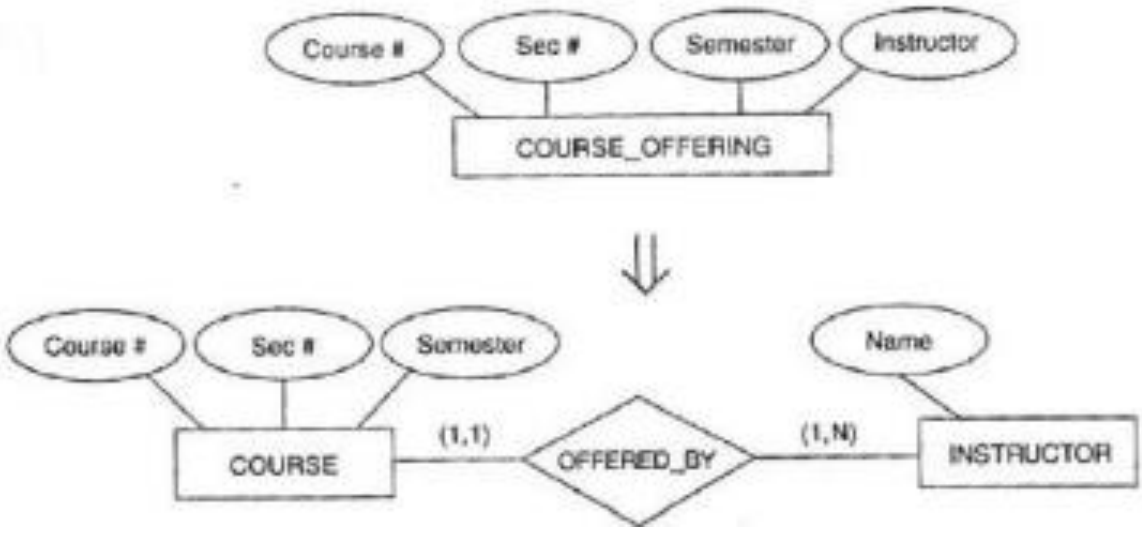
STRATEGI TOP-DOWN

Membangkitkan tipe entiti baru.



Gambar 4. 18 Contoh Tipe Entiti Baru

Dekomposisi tipe entiti ke dalam dua tipe entiti dan relasi

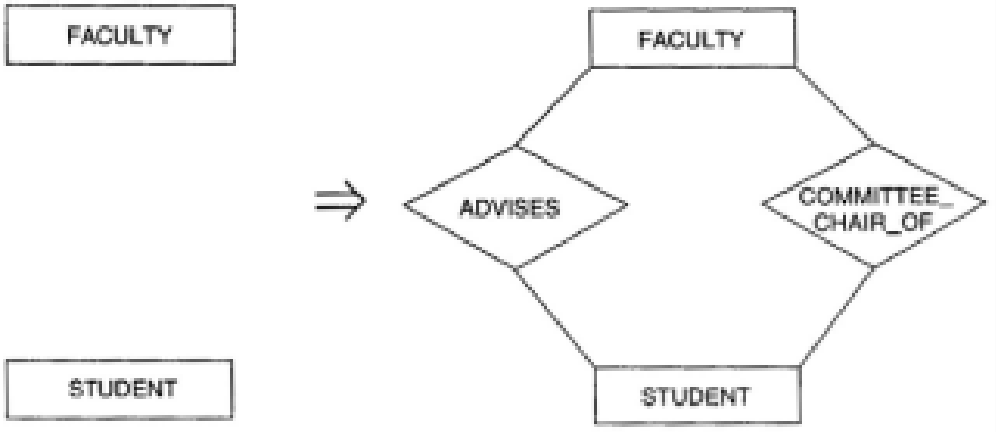


Gambar 4. 19 Contoh Tipe Entiti Dalam Dua Tipe Entiti Relasi

STRATEGI BOTTOM -UP

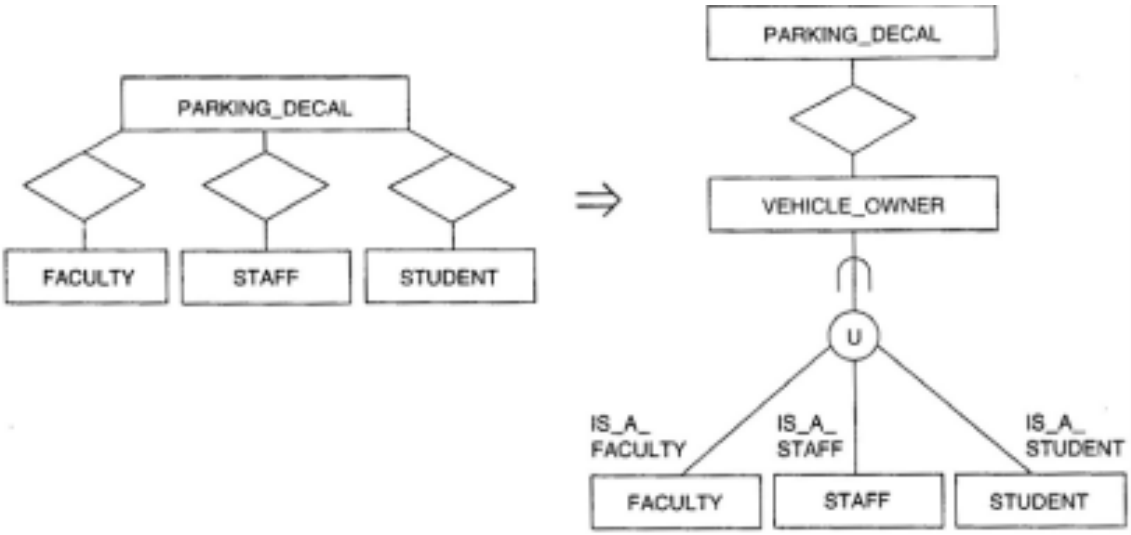
Contoh strategi bottom-up.

Menemukan dan menambah relasi baru.



Gambar 4. 20 Contoh Menemukan dan Menambah Relasi Baru

Menemukan katagori baru (tipe union) dan menghubungkannya.



Gambar 4. 21 Contoh Menemukan Katagori Baru Tipe Union

SKEMA INTEGRASI (VIEW)

Untuk desain database besar→ skema individual → gabungkan ▪ Dibagi menjadi : Identifikasi Korespondensi dan konflik diantara skema antara lain : Naming Conflict, Type Conflict, Domain Conflict, Constraint Conflict. Modifikasi View untuk kesesuaian dengan yg lain. Menggabungkan View. Restrukturisasi

2. Desain Transaksi Dan Aplikasi

Transaksi dan User Interface

- Pada saat basis data didesain, aplikasi dari transaksi utama harus sudah diketahui
- Transaksi-transaksi baru dapat didefinisikan kemudian
- Tentukan karakteristik dari transaksi dan periksa apakah basis data sudah memuat semua informasi untuk melaksanakan transaksi
- Transaksi dapat dibagi dalam 3 bagian yaitu: retrieval, update, mixed
- Tahap 2A dan 2B sebaiknya dilaksanakan secara paralel dengan menggunakan umpan balik agar didapat skema desain dan transaksi yang stabil

Desain Transaksi

Teknik yang umum digunakan adalah mengidentifikasi parameter input/output dan aliran fungsi internal. Transaksi dikelompokkan dalam 3 kategori :

- Retrieval Transaction Untuk menampilkan data ke layar atau untuk produksi pelaporan.
- Update Transaction Untuk memasukkan data baru atau memodifikasi data yang sudah ada pada basis data.
- Mixed Transaction Untuk aplikasi yang komplek yang melakukan retrieval dan update. Contoh : Pemesanan tiket secara online, retrieval transaction → menampilkan daftar semua pesawat, update transaction → booking tempat duduk pada jalur tertentu

Desain User Inter Face

Beberapa aturan pokok dalam merancang User Interface:

- Pemberian nama form jelas, menerangkan kegunaan dari form dan laporan
- Pemberian Intruksi dapat dimengerti
- Pengelompokan secara logik dan pengurutan field
- Tampilan form/report secara visual
- Nama field familiar
- emakaian istilah dan singkatan konsisten
- Penggunaan warna konsisten
- Ruang yang tersedia dan cakupan untuk field pemasukan data
- Perpindahan kurson yang tepat
- Perbaikan kesalahan untuk karakter individual, maupun field secara keseluruhan
- Pesan kesalahan untuk nilai yang tdk diterima
- Fiel pilihan ditandai dengan jelas
- Pesan penjelasan untuk field
- Penanda akhir yang menyatakan proses sudah selesai

TAHAP 3:

PEMILIHAN DBMS

Langkah Utama dalam memilih DBMS : (Connoly)

- Lihat informasi DBMS dari referensi
- Buat daftar 2 atau 3 produk
- Evaluasi produk
- Rekomendasi dan buat reportnya

Bebarapa Fitur Untuk Evaluasi DMBS

Data Difinition	Physical Definition	Accessiblity
Primary key enformcement Foreign key specification Data types available Easy of restructuring Integrity control View mechanism Data dictionary dll	File structured avalaible File structured maintenance Easy of reorganization Indexing Data compression Memory requirements Storage requirements	Multi user Security (access control, authorization mechanism
Transaction Handling	**Utilities**	
Back up & recovery routines Checkpoint facility Logging facility dll	Load & unload facilities Database administration support	

Gambar 4. 22 Contoh Fitur Untuk Evaluasi DMBS

FAKTOR

Faktor dalam Pemilihan DBMS :

- Faktor teknis: berhubungan dengan ketepatan DBMS yang dipilih (tipe DBMS : relational, object relational dll) Struktur penyimpanan, storage, akses path, ketersediaan user interface dan programmer, bahasa query, dll
- Faktor ekonomi: Biaya Software, biaya Hardware, Biaya pembuatan database dan konversi, biaya Maintenance, Personal cost ,training, operasi.
- Faktor Organisasi : Struktur organisasi, Personal yang terbiasa dengan sistem yang terdahulu , Ketersediaan dari service vendor

Faktor Teknis

- DBMS (relational, hirarki, atau jaringan)
- Struktur penyimpan dan akses path yang didukung DBMS
- Ketersediaan antar muka pemakai dan pemrogram, tipe bahasa query tingkat tinggi, ketersediaan alat bantu pengembangan, kemampuan berhubungan dengan DBMS lain melalui media standard
- Pilihan arsitektur yang berhubungan dengan operator clientserver dan lain sebagainya.

FAKTOR EKONOMI

- Software Acquisiton Cost Pembelian perangkat lunak, termasuk pilihan bahasa, pilihan antar muka seperti form, menu dan antar muka Web berbasis GUI, pilihan recovery/backup.
- Maintenance Cost Berhubungan dengan harga layanan pemeliharaan standar dari vendor dan untuk menjaga versi DBMS tetap up to date.
- Hardware Acquisition Cost Perangkat keras baru mungkin diperlukan, seperti memory, terminal, disk drive dan controller baru, atau penyimpan DBMS khusus.
- Database Creation and Conversion Cost Biaya pembuatan sistem basis data dari konversi sistem yang sudah ada ke perangkat lunak DBMS baru.
- Personal cost Akuisisi perangkat lunak DBMS untuk pertama kali oleh organisasi biasanya dilakukan dengan reorganisasi departemen data processing.
- Training cost Karena DBMS biasanya berupa sistem komplek, personal harus ditraining menggunakan dan memprogram DBMS. Training diperlukan pada semua level, termasuk programming, pengembangan aplikasi dan administrasi basis data.
- Operating cost : Biaya operasi lanjutan dari sistem basis data biasanya tidak termasuk dalam evaluasi.

FAKTOR ORGANISASI

- Struktur Data Jika data yang disimpan dalam database mengikuti struktur hirarki, maka suatu jenis hirarki dari DBMS harus dipikirkan.
- Familiarity of Personnel with the System Jika staff programming dalam organisasi familiar dengan DBMS tertentu, dapat mengurangi biaya training dan waktu pembelajaran.
- Availability of Vendor Service Kedengan sistem sangat penting, karena perubahan dari non-DBMS tersediaan asisten vendor dalam pemecahan permasalahan ke lingkungan DBMS kebanyakan membutuhkan bantuan vendor pada awalnya.

TAHAP 4

Pemetaan Model Data (DESAIN BASIS DATA LOGIKA) Membuat skema konseptual dan skema eksernal dalam model data dari DBMS terpilih Proses pemetaan dalam dua bentuk :

- Pemetaan yang Tidak Tergantung pada Sistem (SystemIndependet Mapping) Pada bentuk ini, pemetaan tidak mempertimbangkan karakteristik khusus atau kasus khusus yang diaplikasikan ke implementasi DBMS dari model data.

- Penyesuaian Skema ke DBMS yang Spesifik (Tailoring the Schemas to Specific DBMS) DBMS yang berbeda mengimplementasikan model data dengan menggunakan pemodelah khusus. Hasilnya merupakan pernyataan DDL dari DBMS yang dipilih

TAHAP 5

Desain Database Secara Fisik

- Proses pemilihan struktur penyimpanan dan jalur akses pada file-file basis data untuk mencapai penampilan yang terbaik pada bermacam aplikasi.

- Dirancang spesifikasi-spesifikasi untuk basis data yang disimpan yang berhubungan dengan struktur-struktur penyimpanan fisik, penempatan record dan jalur akses

Beberapa petunjuk dalam pemilihan perancangan basis data secara fisik :

- Waktu Respon: Waktu transaksi basis data untuk menerima respon selama eksekusi. Waktu respon dipengaruhi waktu akses basis data untuk data item yang ditunjuk oleh suatu transaksi.

- Penggunaan Ruang Penyimpanan Jumlah ruang penyimpanan yang digunakan oleh file basis data dan struktur- struktur jalur akses.

- Transaction Throughput Rata-rata jumlah transaksi yang dapat diproses per menit oleh sistem basis data, dan merupakan parameter kritis dari sistem transaksi (misal : digunakan pada pemesanan tempat di pesawat, bank, dll).

Tahap 6

Implementasi Basis Data

- DBA bersama desainer basis data menggunakan pernyataan dalam DDL , SDL (Storage Definition Language) dari DBMS terpilih digunakan untuk membuat skema basisdata dan file basis data (kosong).

- Basis data kemudian dipopulasikan dengan data.

- Jika data diubah dari sistem komputerisasi sebelumnya, rutin konversi diperlukan untuk format kembali data untuk menyimpan ke basis data baru.

- Transaksi basis data harus diimplementasikan dengan aplikasi yang dibuat programmer

- Melakukan uji coba kode porgram dengan perintah DML.

- Jika transaksi siap dan data disimpan ke basis data, tahap rancangan dan implementasi selesai dan tahap operasi dari sistem basis data dimulai.

TAHAP 7

KONVERSI & LOADING DATA

- Tahap ini dilakukan apabila sistem basis data yg ada digantikan sistem basis data baru

- Semua data yg ada ditransfer ke basis data baru & konversi aplikasi yg ada utk basis data baru

TESTING & EVALUASI

- Dilakukan pengujian utk kinerja, integritas, pengaksesan konkuren, keamanan dari basis data

- Dilakukan paralel dg pemrograman aplikasi

- Jika hasil gagal dilakukan maka : Diuji berdasarkan referensi manual, Modifikasi perancangan fisik, Modifikasi perancangan logic, Upgrade atau pengubahan perangkat lunak DBMS & perangkat keras

PENGOPERASIAN & PERAWATAN

- Pengoperasian basis data setelah divalidasi
- Memonitor kinerja sistem, jika tidak sesuai perlu reorganisasi basis data
- Perawatan & upgrade sistem aplikasi basis data jika diperlukan.

Bab 5 Implementasi Sistem Basis Data

5.1 Tahap Implementasi

Pada tahap ini dilakukan implementasi dari perancangan database yang telah dilakukan pada tahap sebelumnya. Berikut ini tampilan implementasi database Aplikasi Pemasaran Online:

Gambar 5.1 adalah tabel Admin yang berfungsi untuk menampung data – data administrasi. Dalam tabel admin terdapat atribute yaitu id_admin, nama_admin, jekel, facebook, email, id_user. Dimana id_admin sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_admin	int(20)			No	0		
	nama_admin	varchar(30)	latin1_swedish_ci		No	None		
	jekel	varchar(20)	latin1_swedish_ci		No	None		
	facebook	varchar(50)	latin1_swedish_ci		No	None		
	email	varchar(50)	latin1_swedish_ci		No			
	id_user	int(10)			No	None		

Gambar 5. 1 Contoh Tabel Admin

Gambar 5.2 adalah potongan script SQL tabel Admin pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saatdiinputkan beberapa atribut dalam tabel admin. Atributnya meliputi id_admin, nama_admin, jekel, facebook, email dan id_user.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `admin` WHERE 1 `id_admin`, `nama_admin`, `jekel`, `facebook`, `email`, `id_user`
```

Gambar 5. 2 Contoh Potongan Script SQL Tabel Admin

Gambar 5.3 adalah tabel barang yang berfungsi untuk menampung data – data barang member. Dalam tabel barang terdapat atribute yaitu id_barang, nama_barang, id_toko, nama_toko, gambar_barang, harga, jenis_barang, stok. Dimana id_barang sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_barang	varchar(20)	latin1_swedish_ci		No	None		
	nama_barang	varchar(50)	latin1_swedish_ci		No	None		
	id_toko	int(20)			No	None		
	nama_toko	varchar(30)	latin1_swedish_ci		No	None		
	gambar_barang	varchar(50)	latin1_swedish_ci		No	None		
	harga	varchar(30)	latin1_swedish_ci		No	None		
	jenis_barang	varchar(30)	latin1_swedish_ci		No	None		
	stok	int(3)			No	None		

Gambar 5. 3 Contoh Tabel Barang

Gambar 5.4 adalah potongan script SQL tabel Barang pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saat diinputkan beberapa atribut dalam tabel barang. Atributnya meliputi id_barang, nama_barang, id_toko, nama_toko, gambar_barang, harga, jenis_barang, stok.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `barang` WHERE 1`id_barang`, `nama_barang`, `id_toko`, `nama_toko`, `gambar_barang`,
`harga`, `jenis_barang`, `stok`
```

Gambar 5. 4 Contoh Potongan Script SQL Tabel Barang

Gambar 5.5 adalah tabel detil pesan yang berfungsi untuk menampung data – data pesanan secara detail. Dalam tabel de teil pesan terdapat atribute yaitu kode_detil_pesan, kode_pesan, id_barang, struk_pesan, tglpesan, jumlahpesan, totalbeli, keterangan. Dimana kode_detil_pesan sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	kode_detil_pesan	int(20)			No	None		
	kode_pesan	int(20)			No	None		
	id_barang	int(20)			No	None		
	struk_pesan	varchar(100)	latin1_swedish_ci		No	None		
	tglpesan	date			No	None		
	jumlahpesan	int(5)			No	None		
	totalbeli	int(10)			No	None		
	keterangan	text	latin1_swedish_ci		No	None		

Gambar 5. 5 Contoh Tabel Detail Pesan

Gambar 5.6 adalah potongan script SQL tabel detil pesan pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saat diinputkan beberapa atribut dalam tabel

detil_pesan. Atributnya meliputi kode_detil_pesan, kode_pesan, id_barang, struk_pesan, tglpesan, jumlahpesan, totalbeli, keterangan.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `detil_pesan` WHERE 1`kode_detil_pesan`, `kode_pesan`, `id_barang`, `struk_pesan`,
`tglpesan`, `jumlahpesan`, `totalbeli`, `keterangan`
```

Gambar 5. 6 Contoh Potongan Script SQL Tabel Detail Pesan

Gambar 5.7 adalah tabel komentar yang berfungsi untuk menampung komentar dari pengunjung maupun member. Dalam tabel komentar terdapat atribute yaitu id_comment, comment, like, id_barang, nama, tanggal. Dimana id_comment sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_comment	int(10)			No	None	auto_increment	
	comment	varchar(100)	latin1_swedish_ci		No	None		
	like	varchar(20)	latin1_swedish_ci		No	None		
	id_barang	int(20)			No	None		
	nama	varchar(40)	latin1_swedish_ci		No	None		
	tanggal	date			No	None		

Gambar 5. 7 Contoh Tabel Komentar

Gambar 5.8 adalah potongan script SQL tabel komentar pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saat diinputkan beberapa atribut dalam tabel komentar. Atributnya meliputi id_comment, comment, like, id_barang, nama, tanggal.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `komentar` WHERE 1`id_comment`, `comment`, `like`, `id_barang`, `nama`, `tanggal`
```

Gambar 5. 8 Contoh Potomgan Script SQL Tabel Komentar

Gambar 5.9 adalah tabel pemesan yang berfungsi untuk menampung data – data pemesan. Dalam tabel pemesan terdapat atribute yaitu kode_pesan, nama_pemesan, alamat, telp, email, pinbb, noidentitas. Dimana kode_pemesan sebagai primary key

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	kode_pesan	int(20)			No	None		
	nama_pemesan	varchar(30)	latin1_swedish_ci		No	None		
	alamat	text	latin1_swedish_ci		No	None		
	telp	varchar(15)	latin1_swedish_ci		No	None		
	email	varchar(40)	latin1_swedish_ci		No	None		
	pinbb	varchar(10)	latin1_swedish_ci		No	None		
	noidentitas	int(40)			No	None		

Gambar 5. 9 Contoh Tabel Pemesan

Gambar 5.10 adalah potongan script SQL tabel pemesanan pada jendela php myAdmin.Script SQL akan menampilkan hasil tabel saat diinputkan beberapa atribut dalam tabel pemesan. Atributnya meliputi kode_pesan, nama_pemesan, alamat, telp, email, pinbb, noidentitas.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `pemesan` WHERE 1 `kode_pesan`, `nama_pemesan`, `alamat`, `telp`, `email`, `pinbb`,
`noidentitas`
```

Gambar 5. 10 Contoh Potongan Script SQL Tabel Pemesan

Gambar 5.11 adalah tabel member_toko yang berfungsi untuk menampung data – data dari member yang mendaftar. Dalam tabel member_toko terdapat atribute yaitu id_toko, nama_toko, alamat_toko, no_hp, nama_pemilik, email, facebook , bbm, id_user. Dimana id_toko sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_toko	int(20)			No	None		
	nama_toko	varchar(30)	latin1_swedish_ci		No	None		
	alamat_toko	varchar(50)	latin1_swedish_ci		No	None		
	no_hp	int(20)			No	None		
	nama_pemilik	varchar(50)	latin1_swedish_ci		No	None		
	email	varchar(50)	latin1_swedish_ci		No	None		
	facebook	varchar(50)	latin1_swedish_ci		No	None		
	bbm	varchar(20)	latin1_swedish_ci		No	None		
	id_user	int(10)			No	None		

Gambar 5. 11 Contoh Tabel Member Toko

Gambar 5.12 adalah potongan script SQL tabel member toko pada jendela php myAdmin. Script SQL akan menamplkan hasil tabel saat diinputkan beberapa atribut dalam tabel

member_toko.Atributnya antara lain id_toko, nama_toko, alamat_toko, no_hp, nama_pemilik, email, facebook , bbm, id_user.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `toko_member` WHERE 1`id_toko`, `nama_toko`, `alamat_toko`, `no_hp`, `nama_pemilik`,
`email`, `facebook`, `bbm`, `id_user`
```

Gambar 5. 12 Contoh Potongan Script SQL Tabel Member Toko

Gambar 5.13 adalah tabel user yang berfungsi untuk menampung data – data user dari sistem. Dalam tabel user terdapat atribute yaitu id_user, username, password, status. Dimana id_user sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_user	int(10)			No	None		
	username	varchar(30)	latin1_swedish_ci		No	None		
	password	varchar(30)	latin1_swedish_ci		No	None		
	status	varchar(20)	latin1_swedish_ci		No	None		

Gambar 5. 13 Contoh Tabel User

Gambar 5.14 adalah potongan script SQL tabel user pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saat diinputkan beberapa atribut dalam tabel user. Atributnya antara lain id_user, username, password, status.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `user` WHERE 1`id_user`, `username`, `password`, `status`
```

Gambar 5. 14 Contoh Potongan Script SQL Tabel User

Gambar 5.15 adalah tabel suka yang berfungsi untuk menampung data – data jumlah like pengunjung pada barang. Dalam tabel suka terdapat atribute yaitu id, id_admin. Dimana id sebagai primary key.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id	int(10)			No	None		
	id_barang	varchar(20)	latin1_swedish_ci		No	None		

Gambar 5. 15 Contoh Tabel Suka

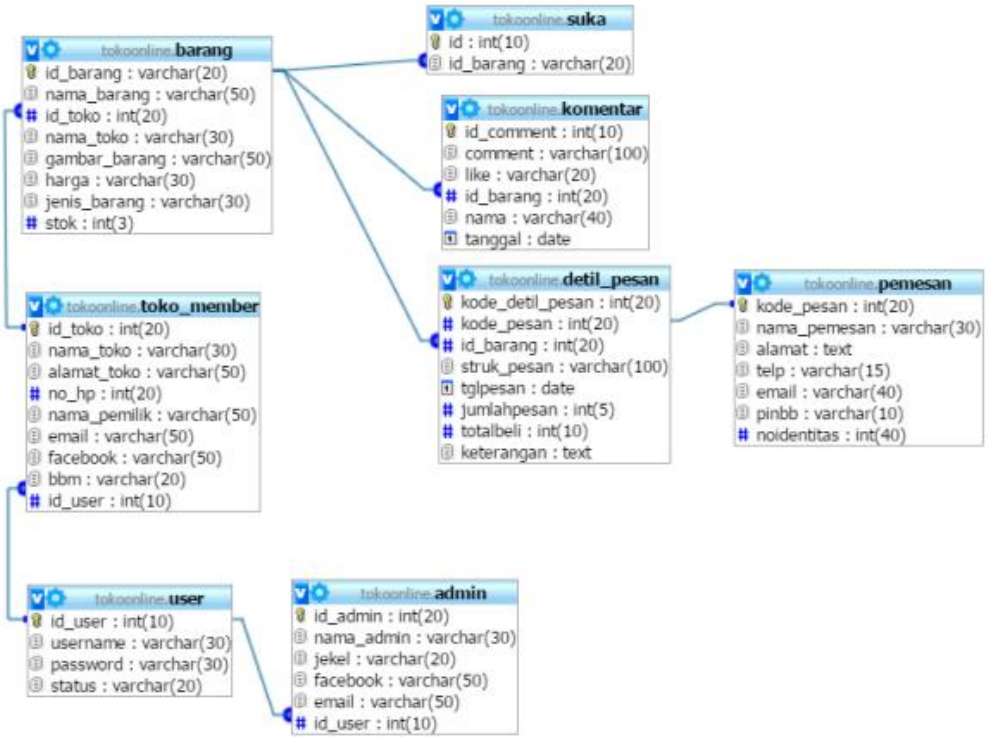
Gambar 5.16 adalah potongan script SQL tabel suka pada jendela php myAdmin. Script SQL akan menampilkan hasil tabel saatdiinputkan beberapa atribut didalamnya. Atributnya antara lain id, id_admin.

Run SQL query/queries on database tokoonline:

```
SELECT * FROM `suka` WHERE 1`id`, `id_barang`
```

Gambar 5. 16 Contoh Potongan Script SQL Tabel Suka

Setelah tabel – tabel yang telah dirancang diimplementasikan, maka selanjutnya adalah penerapan relasi antar tabel database yang meliputi tabel barang, toko_member, user, admin, suka, komentar, detil_pesan, pemesan seperti pada gambar 5.17 :



Gambar 5. 17 Contoh Relasi Tabel Database

5.2 Pemilihan Perangkat Lunak Basis Data

Pemilihan perangkat lunak basis data merupakan langkah kritis dalam pengembangan sistem informasi yang sukses. Dalam dunia yang didorong oleh data seperti saat ini, pemilihan perangkat lunak basis data yang tepat dapat memiliki dampak besar pada kinerja, keamanan, dan keseluruhan efisiensi sistem. Proses ini melibatkan pertimbangan yang matang terhadap berbagai faktor yang mempengaruhi keberhasilan implementasi dan penggunaan basis data. Berikut adalah beberapa aspek penting yang perlu dipertimbangkan dalam pemilihan perangkat lunak basis data:

- **Kebutuhan Bisnis yang Spesifik**: Pemilihan perangkat lunak basis data harus selaras dengan kebutuhan bisnis dan tujuan organisasi secara keseluruhan. Penting untuk memahami kebutuhan fungsional dan non-fungsional sistem secara menyeluruh. Apakah organisasi memerlukan basis data relasional tradisional untuk mendukung transaksi bisnis sehari-hari, atau apakah mereka membutuhkan solusi NoSQL untuk mengelola data yang tidak terstruktur?

- **Skalabilitas dan Kinerja**: Kemampuan perangkat lunak basis data untuk menangani beban kerja yang tumbuh seiring waktu sangat penting. Pertimbangkan faktor-faktor seperti kinerja, skalabilitas horizontal dan vertikal, serta kemampuan untuk menangani volume data yang besar. Solusi yang dipilih harus dapat memberikan kinerja yang optimal bahkan saat sistem berkembang dan memproses data dalam skala yang besar.

- **Keamanan Data**: Dalam era di mana keamanan data menjadi perhatian utama, penting untuk memilih perangkat lunak basis data yang menyediakan fitur keamanan yang kuat. Ini termasuk enkripsi data, manajemen hak akses pengguna yang canggih, dan fitur auditabilitas yang memungkinkan pelacakan aktivitas pengguna.

- **Biaya dan Model Lisensi**: Biaya akuisisi, implementasi, dan pemeliharaan perangkat lunak basis data harus menjadi pertimbangan utama. Tinjau model lisensi, biaya dukungan, dan biaya infrastruktur yang terkait dengan solusi yang dipilih. Pilihlah solusi yang memberikan nilai terbaik sesuai dengan anggaran organisasi.
- **Dukungan dan Komunitas**: Tingkat dukungan teknis yang disediakan oleh vendor perangkat lunak basis data juga merupakan faktor penting. Penting untuk memilih vendor yang dapat memberikan dukungan teknis yang andal dan responsif. Selain itu, adanya komunitas pengguna yang aktif juga dapat menjadi sumber daya berharga dalam menyelesaikan masalah dan berbagi pengetahuan.

- **Integrasi dengan Teknologi Lain**: Perangkat lunak basis data harus dapat berintegrasi dengan teknologi lain yang sudah ada dalam infrastruktur IT organisasi. Pastikan solusi yang dipilih kompatibel dengan platform dan alat lain yang digunakan, serta memiliki dukungan untuk standar integrasi yang umum digunakan seperti JDBC/ODBC.

- **Ketersediaan dan Ketahanan**: Ketersediaan dan ketahanan sistem basis data sangat penting, terutama dalam lingkungan bisnis yang 24/7. Tinjau fitur-fitur seperti replikasi data, clustering, dan pemulihan bencana untuk memastikan bahwa data tetap tersedia bahkan dalam situasi darurat.

- **8. Evaluasi dan Uji Coba**: Terakhir, sebelum membuat keputusan akhir, lakukan evaluasi dan uji coba terhadap beberapa solusi yang dipertimbangkan. Lakukan analisis mendalam terhadap fitur, kinerja, dan keandalan masing-masing solusi, serta cari umpan balik dari tim teknis dan pengguna akhir.

Dengan mempertimbangkan faktor-faktor ini secara cermat, organisasi dapat membuat keputusan yang tepat dalam pemilihan perangkat lunak basis data yang sesuai dengan kebutuhan bisnis dan teknis mereka. Langkah ini merupakan langkah krusial dalam memastikan bahwa sistem informasi berjalan dengan lancar dan efisien dalam jangka panjang, serta mampu mendukung pertumbuhan dan transformasi organisasi.

5.3 Instalasi dan Konfigurasi Perangkat Lunak Basis Data

Instalasi dan konfigurasi perangkat lunak basis data adalah langkah awal yang penting dalam mempersiapkan lingkungan pengembangan atau produksi untuk aplikasi yang bergantung pada penyimpanan dan pengelolaan data. Proses ini melibatkan beberapa tahapan yang membutuhkan pemahaman mendalam tentang persyaratan sistem, konfigurasi jaringan, dan spesifikasi perangkat keras. Berikut adalah panduan umum tentang instalasi dan konfigurasi perangkat lunak basis data:

1. Pemilihan Versi dan Edisi: Pertama-tama, tentukan versi dan edisi perangkat lunak basis data yang akan diinstal. Pilih versi yang sesuai dengan kebutuhan aplikasi dan kompatibel dengan sistem operasi yang digunakan.

2. Persyaratan Sistem: Pastikan bahwa sistem memenuhi persyaratan minimum untuk menjalankan perangkat lunak basis data dengan baik. Ini termasuk ruang penyimpanan yang cukup, RAM yang memadai, dan prosesor yang sesuai.

3. Pemilihan Platform: Tentukan platform atau sistem operasi tempat perangkat lunak basis data akan diinstal. Pastikan perangkat lunak yang dipilih kompatibel dengan sistem operasi yang digunakan, apakah itu Windows, Linux, atau platform lainnya.

4. Unduh dan Instalasi: Unduh paket instalasi perangkat lunak basis data dari situs web resmi vendor atau penyedia. Ikuti panduan instalasi yang disediakan oleh vendor untuk menginstal perangkat lunak dengan benar. Pastikan untuk memperhatikan opsi instalasi yang tersedia, termasuk lokasi instalasi, pengaturan default, dan opsi tambahan.

5. Konfigurasi Awal: Setelah instalasi selesai, lakukan konfigurasi awal perangkat lunak basis data. Ini termasuk pengaturan parameter konfigurasi seperti direktori data, port komunikasi, dan konfigurasi keamanan. Pastikan untuk memeriksa dokumentasi perangkat lunak untuk panduan konfigurasi yang tepat.

6. Pengujian Koneksi: Setelah konfigurasi awal selesai, uji koneksi ke perangkat lunak basis data untuk memastikan bahwa instalasi berhasil dan dapat diakses dari aplikasi atau perangkat lunak lain yang membutuhkan akses ke basis data.

7. Pengaturan Keamanan: Perangkat lunak basis data umumnya menyediakan berbagai fitur keamanan untuk melindungi data sensitif. Konfigurasikan opsi keamanan yang sesuai, seperti pengaturan kata sandi, akses pengguna, dan enkripsi data.

8. Pemantauan dan Pemeliharaan: Setelah instalasi dan konfigurasi selesai, pastikan untuk memantau kinerja sistem secara berkala dan melakukan pemeliharaan rutin, seperti pemutakhiran perangkat lunak dan pembersihan data yang tidak perlu.

9. Dokumentasi: Terakhir, penting untuk mendokumentasikan proses instalasi dan konfigurasi secara rinci. Ini akan membantu dalam pemeliharaan sistem dan pemecahan masalah di masa mendatang.

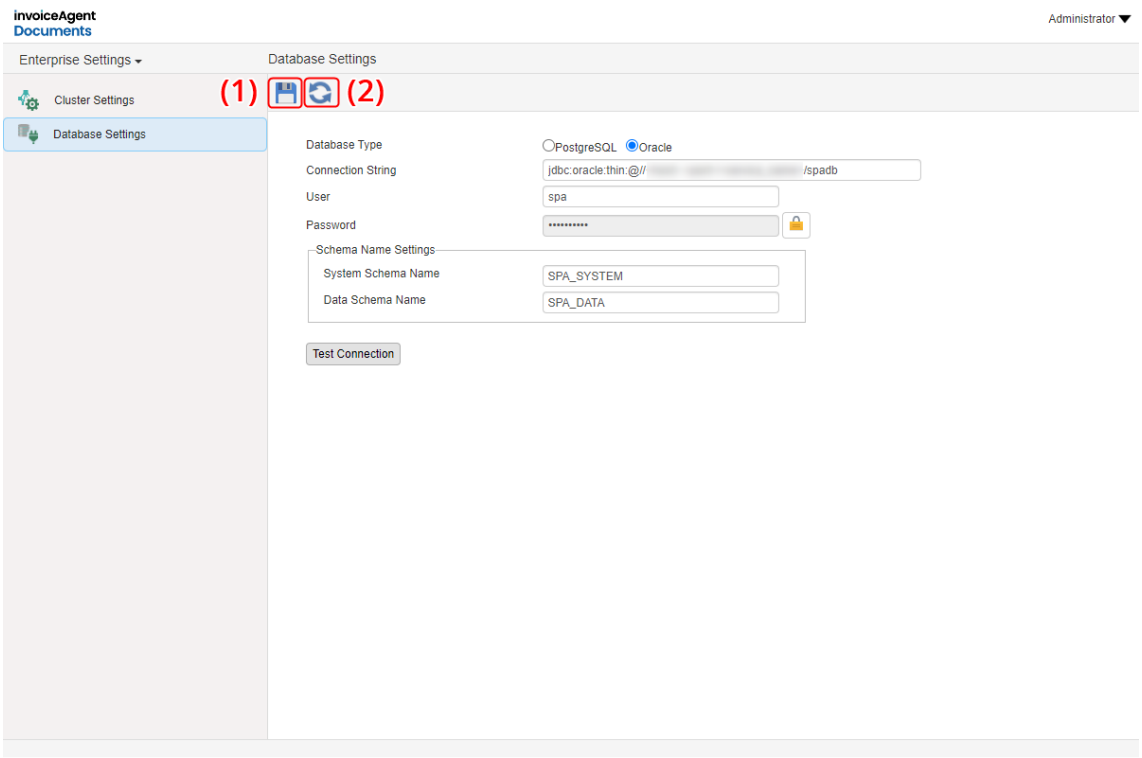
Penting untuk diingat bahwa proses instalasi dan konfigurasi perangkat lunak basis data dapat bervariasi tergantung pada perangkat lunak yang dipilih dan lingkungan spesifik organisasi. Oleh karena itu, pastikan untuk merujuk ke dokumentasi resmi perangkat lunak dan memperoleh bantuan dari tim teknis atau vendor jika diperlukan.

5.4 Pengaturan basis data

Pengaturan Basis Data adalah layar untuk mengatur database yang akan digunakan dalam cluster olehinvoiceAgent Documents server yang pertama kali bergabung dengan cluster. Item pengaturan bervariasi tergantung pada tipe database.

Untuk menampilkannya, pilih kategori pengaturan Pengaturan Perusahaan - Pengaturan Basis Data di layar manajemen.

Item dalam "Pengaturan Basis Data"



Gambar 5. 18 Contoh Pengaturan Basis Data

NO	Item	Keterangan
(1)	**Menyimpan** tombol	Menyimpan pengaturan. Setelah Anda menyimpan informasi database, tombol simpan akan dinonaktifkan.
(2)	**Menyegarkan** tombol	Memperbarui tampilan ke status terbaru. Jika Anda mengkliknya saat memasukkan informasi database, data yang dimasukkan akan dibuang.
	Tipe Basis Data	Tentukan jenis database yang akan digunakan. • **PostgreSQL** • **Peramal**
	String koneksi	Masukkan string koneksi untuk database. Gunakan format berikut: • Untuk PostgreSQL

NO	Item	Keterangan
		jdbc:postgresql://<database_server_name_or_IP_address>:<database_port_number>/spadb • Untuk Oracle jdbc:Oracle:thin:@//<database_server_name_or_IP_address>:<database_port_number>/<database_service_name>
	Pengguna	Masukkan pengguna untuk terhubung ke database.
	Kata sandi	Masukkan kata sandi pengguna untuk terhubung ke database.
	Pengaturan Nama Skema	Setel hanya ketika " **Peramal** " ditentukan dalam **Tipe Basis Data** . • **Nama Skema Sistem** Masukkan nama skema sistem dari skema yang dibuat untukinvoiceAgent Documents . • **Nama Skema Data** Masukkan nama skema data dari skema yang dibuat untukinvoiceAgent Documents .
	Tes koneksi tombol	Jika Anda mengkliknya setelah memasukkan item di atas, Anda dapat memeriksa apakah Anda dapat membangun koneksi dengan informasi yang dimasukkan.

Tabel 5. 1 Item dan Keterangan

Bab 6 Operasi Sistem Basis Data

6.1 Pengelolaan Data

Pengelolaan data merupakan aspek yang sangat penting dalam dunia modern yang didorong oleh informasi. Ini mencakup serangkaian proses dan praktik yang bertujuan untuk mengumpulkan, menyimpan, mengatur, mengamankan, dan memelihara data agar dapat digunakan secara efektif oleh organisasi. Pengelolaan data yang baik memastikan bahwa data tersedia, akurat, terpercaya, dan dapat diakses dengan mudah oleh mereka yang membutuhkannya. Berikut adalah beberapa aspek penting yang perlu dipertimbangkan dalam pengelolaan data:

- **Pengumpulan Data**: Proses pengelolaan data dimulai dengan pengumpulan data dari berbagai sumber, baik itu sumber internal maupun eksternal. Ini bisa meliputi data transaksi, data pelanggan, data produksi, data penjualan, dan banyak lagi. Penting untuk memastikan bahwa data yang dikumpulkan relevan, akurat, dan lengkap.

- **Penyimpanan Data**: Setelah data dikumpulkan, langkah selanjutnya adalah menyimpannya dalam basis data atau penyimpanan lain yang sesuai. Penting untuk memilih metode penyimpanan yang tepat, termasuk basis data relasional, basis data NoSQL, data warehousing, atau penyimpanan berbasis cloud, tergantung pada kebutuhan dan karakteristik data.

- **Organisasi Data**: Data perlu diatur dan diindeks agar dapat diakses dengan efisien. Ini melibatkan pembuatan struktur data yang terstruktur dan hierarkis, termasuk penggunaan skema, tabel, kolom, dan indeks dalam basis data relasional, serta pengaturan kategori dan atribut dalam basis data NoSQL.

- **Keamanan Data**: Keamanan data adalah aspek yang sangat penting dalam pengelolaan data. Ini melibatkan penerapan kebijakan keamanan yang tepat, seperti pengaturan hak akses pengguna, enkripsi data, pemantauan aktivitas pengguna, dan pemulihan bencana untuk melindungi data dari ancaman eksternal dan internal.

- **Pemeliharaan Data**: Data perlu dipelihara secara teratur untuk memastikan keakuratan, konsistensi, dan integritasnya. Ini meliputi pembaruan data, pembersihan data yang tidak relevan, penghapusan data yang sudah tidak terpakai, serta pemantauan kualitas data secara berkala.

- **Analisis Data**: Pengelolaan data juga mencakup analisis data untuk mendapatkan wawasan dan informasi yang berharga dari data yang tersedia. Ini melibatkan penggunaan alat dan teknik analisis data seperti OLAP (Online Analytical Processing), data mining, analisis prediktif, dan visualisasi data untuk mengeksplorasi tren, pola, dan hubungan dalam data.

- **Kepatuhan Regulasi**: Dalam lingkungan yang diatur ketat seperti sekarang, penting untuk mematuhi regulasi dan kebijakan privasi data yang berlaku. Pengelolaan data harus memperhatikan peraturan seperti GDPR, HIPAA, atau CCPA untuk melindungi data pribadi dan mematuhi persyaratan hukum.

- **Inovasi Data**: Pengelolaan data yang efektif juga memungkinkan organisasi untuk mengambil keuntungan dari inovasi teknologi data baru seperti Big Data, kecerdasan buatan (AI), Internet of Things (IoT), dan analisis real-time untuk menciptakan nilai tambah dan memperoleh keunggulan kompetitif.

Dengan memahami dan menerapkan praktik pengelolaan data yang baik, organisasi dapat memaksimalkan nilai dari data mereka, meningkatkan efisiensi operasional, dan mengambil keputusan yang lebih baik berdasarkan informasi yang akurat dan relevan. Ini adalah bagian yang krusial dari strategi bisnis modern dan dapat menjadi pendorong utama kesuksesan organisasi di era digital.

6.2 Pencadangan dan Pemulihan Basis Data

Pengenalan yang luar biasa ini membutuhkan latar belakang, jadi mari kita selesaikan: Kehilangan data atau waktu henti yang tidak terduga akan merugikan organisasi. Beberapa kerugian bisa melumpuhkan, bahkan menjadi lonceng kematian bisnis. Jadi, penting untuk mengetahui (dan tidak mengabaikan) titik masalahnya. Ada yang tersembunyi, namun ada pula yang terlihat jelas. Secara umum, berikut adalah titik-titik potensi masalah:

Kegagalan perangkat keras: Cacat produksi, keausan, kerusakan fisik, dan faktor lainnya dapat menyebabkan kegagalan perangkat keras. Lonjakan listrik, pemadaman listrik, dan kondisi yang buruk (misalnya panas) dapat merusak komponen perangkat keras dan menyebabkan hilangnya data.

Kegagalan perangkat lunak: Aplikasi perangkat lunak bisa menjadi rentan, atau bisa crash sama sekali. Tanpa mekanisme pencadangan data, dapat terjadi kehilangan data atau downtime sistem. Lebih buruk lagi, seluruh sistem operasi bisa crash, yang juga mengakibatkan hilangnya data.

Kesalahan manusia: Konfigurasi yang salah adalah penyebab umum kegagalan perangkat keras dan perangkat lunak. Demikian pula, penghapusan yang tidak disengaja adalah penyebabnya.

Dalam ketiga kasus tersebut, kegagalan dalam mencadangkan data secara rutin dapat mengakibatkan hilangnya data secara signifikan jika terjadi bencana. Prosedur pencadangan yang solid harus ada.

Jenis dan strategi cadangan

Semuanya dimulai dengan memilih strategi, dan itu bergantung pada faktor-faktor seperti penggunaan dan pentingnya data bagi bisnis Anda, tujuan waktu pemulihan (RTO), dan anggaran Anda. Tergantung pada apa yang dibutuhkan, berikut adalah beberapa jenis, strategi, dan pertimbangan pencadangan database yang umum:

Pencadangan penuh vs. pencadangan tambahan: Paling cocok untuk database yang lebih kecil atau yang tidak mengalami banyak perubahan, pencadangan lengkap mencakup titik pemulihan lengkap. Meskipun hal ini dapat meredakan masalah keamanan, pencadangan penuh dapat memakan waktu dan mahal. Pencadangan tambahan, yang lebih cepat dan memerlukan lebih sedikit penyimpanan dibandingkan pencadangan penuh, menangkap perubahan yang dibuat sejak pencadangan sebelumnya. Cocok untuk database dengan tingkat perubahan sedang. (Untuk penjelasan lebih rinci, baca Cadangan Penuh vs. Inkremental vs. Diferensial: Membandingkan Jenis Cadangan .)

Pencadangan panas vs. pencadangan dingin: Pencadangan panas — digunakan untuk meminimalkan waktu henti dan memastikan ketersediaan data untuk aplikasi penting — memungkinkan Anda membuat salinan database saat sistem masih aktif melayani permintaan pengguna dan memproses transaksi. Dalam pencadangan dingin, database dibuat offline. Hal ini disarankan hanya dalam skenario tertentu, seperti pemeliharaan selama jam-jam penggunaan rendah.

Memilih frekuensi pencadangan yang tepat: Frekuensi yang tepat, tentu saja, akan bervariasi dari satu organisasi ke organisasi lainnya. Jika Anda tidak mampu kehilangan banyak data, Anda memerlukan pencadangan yang lebih sering, bahkan mungkin solusi perlindungan data yang berkelanjutan. Kecepatan perubahan data dalam database Anda juga merupakan faktor penting. Untuk menerapkan perubahan terbaru, database dengan aktivitas penulisan tinggi mungkin memerlukan pencadangan yang lebih sering.

Pencadangan di luar lokasi: Ini melibatkan penyimpanan cadangan secara fisik di lokasi aman di suatu tempat selain pusat data utama. Ini lebih mahal, namun pencadangan di luar lokasi melindungi terhadap bencana di seluruh lokasi.

Dengan strategi backup yang tepat , Anda akan lebih mampu mencapai keberlangsungan bisnis tersebut di atas. Namun, bencana mungkin saja terjadi, jadi mari kita lihat juga pemulihannya.

Strategi dan pengujian pemulihan bencana

Kita mungkin berpikir bahwa bencana akan menimbulkan kerugian, namun dengan perencanaan yang tepat, kerugian tersebut bisa sangat kecil sehingga lebih bersifat "kecelakaan" daripada "buruk". Dengan adanya perencanaan tersebut, Anda dapat melihatnya sebagai berikut: Pencadangan menghasilkan pemulihan bencana.

Strategi pemulihan bencana (DR) sangat penting untuk memastikan integritas, ketersediaan, dan keandalan data, terutama jika terjadi kegagalan atau kesalahan yang tidak terduga. Strategi tersebut membantu memulihkan database ke keadaan yang konsisten dan dapat digunakan. Rencana pemulihan bencana bisa sesederhana penggunaan prosedur pencadangan dan pemulihan, atau bisa juga rumit, bergantung pada RTO dan tujuan titik pemulihan (RPO). Elemen kunci dari rencana pemulihan database yang komprehensif mencakup pemulihan point-in-time, ketersediaan tinggi dan failover, replikasi, dan lainnya (kami akan menguraikannya sedikit).

Apa pun rencana yang Anda ambil, Anda harus mengujinya. Tidak boleh ada jalan pintas dalam pengujian, yang harus mencakup:

Pengujian cadangan: Uji untuk memastikan bahwa data memang dapat dipulihkan dari cadangan. Ini termasuk cadangan data dan log.

Pengujian failover: Jika Anda memiliki pengaturan ketersediaan tinggi atau pusat data sekunder, uji untuk memastikan proses failover beralih ke server database cadangan.

Pengujian aplikasi: Uji untuk memastikan aplikasi berfungsi dengan benar setelah pemulihan.

Konsistensi data: Verifikasi bahwa konsistensi data antara sistem primer dan sekunder akan ada setelah pemulihan.

Pengujian aplikasi: Uji untuk memastikan aplikasi berfungsi dengan benar setelah pemulihan.

Alat pencadangan dan pemulihan

Sekarang setelah kita membahas beberapa strategi, inilah saat yang tepat untuk melihat beberapa alat untuk mewujudkan rencana. Ada terlalu banyak di luar sana untuk memberikan

daftar lengkap, jadi kami hanya akan menyebutkan beberapa opsi penting. Ini jelas bukan penjelasan rinci; itu hanya perkenalan:

MySQL Enterprise Backup: Perangkat lunak ini mendukung pencadangan panas dan pencadangan tambahan.

Oracle Recovery Manager (RMAN): Ini memungkinkan pengguna database Oracle untuk melakukan pencadangan penuh, tambahan, dan diferensial. Alat ini juga menyediakan pemulihan point-in-time.

SQL Server Management Studio: Perangkat lunak ini mencakup alat pencadangan dan pemulihan untuk digunakan dengan Microsoft SQL Server.

Pencadangan MongoDB Atlas: Layanan cloud yang dihosting ini menawarkan pencadangan atau snapshot berkelanjutan untuk pemulihan tepat waktu, serta pencadangan tambahan.

MongoDB Cloud Manager: Layanan yang dihosting ini, yang menggunakan antarmuka pengguna grafis, mendukung pencadangan dan pemulihan kumpulan replika dan cluster yang dipecah.

MongoDB Ops Manager: Tersedia dengan langganan Enterprise Advanced, ini adalah alat lokal yang menyediakan perangkat lunak cadangan dan fitur-fitur yang mirip dengan Cloud Manager.

IBM Data Studio: Ini menyediakan alat berbasis grafis dan baris perintah untuk pencadangan dan pemulihan dalam basis data IBM Db2.

Commvault: Perangkat lunak manajemen dan perlindungan data ini mencakup fitur untuk pencadangan dan pemulihan. Ini digunakan untuk membantu memastikan fungsionalitas perangkat keras, perangkat lunak, dan aplikasi.

Semua opsi di atas memiliki atribut tingkat perusahaan, namun tidak satu pun yang benar-benar open source. Jadi mari kita bahas beberapa opsi yang bersifat open source.

Solusi Percona open source untuk pencadangan dan pemulihan bencana

Saat Anda membaca ulasan tentang apa yang tersedia untuk solusi pencadangan dan pemulihan bencana, Anda cenderung melihat kata-kata cemerlang tentang perangkat lunak dengan nama besar. Namun Anda tidak melihat peringatan bahwa perangkat lunak tersebut merupakan hak milik (paling sering memang demikian); Anda hanya melihat petunjuk, seperti “harga awal”. Dan Anda tentu tidak melihat peringatan bahwa vendor lock-in mungkin mengintai .

Di tengah opsi, Anda melihat Percona di atau dekat bagian atas daftar dan ulasan . Namun dengan Percona, ada perbedaannya. Tidak ada tanda-tanda penguncian kepemilikan karena tidak

ada. Perangkat lunak ini benar-benar open source. Solusi Percona, yang didukung penuh dan berkelas perusahaan, meliputi:

Percona Backup untuk MongoDB : Percona Backup untuk MongoDB adalah solusi terdistribusi dan berdampak rendah untuk pencadangan klaster MongoDB yang konsisten, termasuk dukungan sharding. Ini memungkinkan Anda membuat pencadangan dan pemulihan yang logis, fisik, inkremental, dan selektif. Selain itu, fungsionalitas pemulihan point-in-time memungkinkan Anda memulihkan database ke stempel waktu tertentu.

Percona XtraBackup : Ini adalah solusi pencadangan online lengkap dan gratis untuk semua versi Percona Server untuk MySQL, MySQL, dan MariaDB. Percona XtraBackup melakukan pencadangan online tanpa pemblokiran, terkompresi ketat, dan sangat aman pada sistem transaksional. Percona XtraBackup adalah satu-satunya perangkat lunak hot backup MySQL gratis dan open source yang melakukan pencadangan non-pemblokiran untuk database InnoDB dan XtraDB.

Distribusi Percona untuk PostgreSQL : Distribusi PostgreSQL siap produksi ini mencakup pgBackRest , solusi pencadangan dan pemulihan sumber terbuka yang memungkinkan pencadangan penuh, pencadangan tambahan, dan pencadangan diferensial dalam database PostgreSQL. Perangkat ini juga mendukung ketersediaan tinggi dan pemulihan bencana melalui proksi Patroni, pg_bouncer, dan HA.

Elemen penting dari pencadangan dan pemulihan basis data

Sekarang, mari perkenalkan beberapa elemen solusi pencadangan dan pemulihan yang ideal. (Masih banyak lagi yang akan datang di bagian berikutnya, tapi keduanya sangat besar, sebenarnya sangat besar.)

Replikasi basis data

Redundansi data adalah landasan strategi pemulihan bencana, dan untuk mencapainya, Anda harus memiliki mekanisme replikasi. Bergantung pada kasus penggunaannya, hal ini dapat berarti replikasi sinkron atau replikasi asinkron.

Dalam replikasi sinkron , data ditulis ke database utama. Data kemudian disalin ke satu atau lebih database replika. Basis data utama menunggu pengakuan dari replika sebelum melanjutkan transaksi ke aplikasi. Artinya data di database utama dan replika selalu sinkron. (Lihat whitepaper Percona yang menunjukkan aksi replikasi sinkron .)

Dalam replikasi asinkron , data ditulis ke database utama, namun pengakuan ke aplikasi terjadi sebelum data direplikasi ke database sekunder. Hal ini mengakibatkan penundaan antara

waktu penulisan data ke data primer dan saat data tersebut muncul di replika. Konsistensi data real-time tidak dijamin.

Pemulihan tepat waktu

Dengan pemulihan point-in-time, database dipulihkan ke momen tertentu, bukan waktu pencadangan terbaru. PITR sangat penting dalam situasi ketika integritas dan konsistensi data tidak dapat dikompromikan. Sistem keuangan dan aplikasi bisnis penting sangat bergantung pada PITR. Selain melindungi keakuratan data dan membatasi kehilangan data, PITR dapat membantu persyaratan audit dan kepatuhan dengan menyediakan catatan perubahan ke database.

Komponen umum arsitektur DR dan HA

Berdasarkan definisinya, terdapat perbedaan antara ketersediaan tinggi (HA) dan pemulihan bencana (DR). Ketersediaan tinggi difokuskan untuk mencegah downtime dan memastikan database tetap tersedia; pemulihan bencana difokuskan pada pemulihan dari peristiwa bencana dan meminimalkan dampak negatif terhadap bisnis. Ketersediaan tinggi biasanya melibatkan perangkat keras, perangkat lunak, aplikasi, dan komponen jaringan yang berlebihan yang dapat dengan cepat mengambil alih jika komponen utama gagal; pemulihan bencana biasanya melibatkan pencadangan rutin, replikasi ke situs sekunder, dan rencana pemulihan yang jelas dengan langkah-langkah yang harus diambil jika terjadi bencana.

Meskipun penekanan dan konfigurasinya mungkin berbeda-beda tergantung pada apakah fokusnya adalah pada ketersediaan tinggi atau pemulihan bencana, terdapat komponen-komponen yang sama. Beberapa diantaranya adalah:

Perangkat keras redundan: HA dan DR menggunakan komponen perangkat keras redundan, termasuk server dan perangkat penyimpanan. Redundansi memastikan bahwa jika satu komponen gagal, ada komponen lain yang menggantikannya. Hal ini membantu meminimalkan waktu henti.

Pengelompokan: Di HA, pengelompokan membantu memastikan bahwa ada server basis data yang berlebihan; jika yang satu gagal, yang lain bisa mengambil alih. Hal ini meminimalkan downtime saat terjadi bencana atau kegagalan perangkat keras. Untuk pemulihan bencana, pengelompokan dapat digunakan untuk memelihara salinan database yang tersinkronisasi di lokasi berbeda. Replikasi database, pengiriman log, atau pencerminan data sinkron dapat digunakan untuk tujuan DR.

Penyeimbangan beban: Penyeimbang beban mendistribusikan lalu lintas secara merata ke beberapa server atau pusat data. Di HA, penyeimbang beban membantu memastikan tidak ada

satu server pun yang kewalahan. Di DR, penyeimbang beban merutekan lalu lintas ke pusat data sekunder ketika terjadi failover.

Sistem cadangan: Arsitektur HA dan DR memiliki sistem cadangan. Penyiapan HA cenderung menggunakan server cadangan dalam pusat data yang sama. Penyiapan DR memiliki pusat data cadangan di lokasi berbeda.

Pemantauan dan peringatan: Pemantauan berkelanjutan terhadap kesehatan dan kinerja sistem sangat penting bagi HA dan DR. Peringatan otomatis disiapkan untuk memberi tahu administrator tentang masalah apa pun yang memerlukan perhatian.

Apa yang dipertaruhkan dan apa yang harus dilakukan

Kini setelah kita mendalami lebih dalam komponen solusi pencadangan dan pemulihan bencana, serta melihat ketersediaan tinggi, mari kita perluas kegunaan solusi pencadangan dan pemulihan bencana. Kami juga akan membicarakan apa yang bisa terjadi jika Anda tidak memiliki rencana yang matang, jika Anda mengambil jalan pintas, atau jika Anda menyerahkan semuanya ke vendor yang salah.

Pengarsipan dan penyimpanan data penting

Kita telah membahas sifat penting pencadangan dan pemulihan dalam kelangsungan bisnis. Terkait dengan itu, dengan memisahkan data historis dari data operasional, pengarsipan membantu Anda mengelola pertumbuhan data, menjaga kepatuhan, dan mengoptimalkan proses pencadangan. Demikian pula, menetapkan kebijakan penyimpanan yang jelas untuk pencadangan dan arsip sangat penting untuk menyeimbangkan kebutuhan pemulihan data dengan efisiensi pengelolaan data dan persyaratan kepatuhan.

Dan sifat penting dari kepatuhan tidak dapat cukup ditekankan. Kegagalan untuk mematuhi persyaratan hukum dapat mengakibatkan sanksi moneter dan bahkan pidana, kerusakan reputasi, dan hilangnya integritas data. Berikut beberapa peraturan tersebut:

Peraturan Perlindungan Data Umum (GDPR): Selain menetapkan bahwa organisasi harus memiliki dasar hukum untuk memproses data pribadi, peraturan ini mencakup pedoman kebijakan penyimpanan dan penghapusan data.

Sarbanes-Oxley Act: SOX, undang-undang federal AS, mewajibkan perusahaan untuk memiliki basis data pemulihan bencana dan rencana kelangsungan bisnis. Tujuannya adalah untuk memastikan ketersediaan dan integritas data keuangan di tengah kejadian yang tidak terduga.

Standar Keamanan Data Industri Kartu Pembayaran: PCI DSS mengamanatkan bahwa organisasi harus mencadangkan data secara berkala, termasuk data kartu pembayaran penting, sehingga ketersediaan dan integritas data tetap terjaga. PCI DSS juga menetapkan langkah-langkah untuk merespons pelanggaran data dan insiden keamanan lainnya, termasuk cara memulihkan layanan dan data setelah bencana.

Undang-Undang Privasi Konsumen California: Mirip dengan GDPR, CCPA juga mencakup mandat terkait kebijakan penyimpanan dan penghapusan data.

Peraturan regional: Bergantung pada lokasi Anda dan siapa pelanggan Anda, organisasi Anda mungkin harus mematuhi mandat privasi, pengarsipan, dan penyimpanan sebagaimana dijelaskan dalam HIPAA (layanan kesehatan), FERPA (pendidikan), PIPEDA (Kanada), dan peraturan lainnya.

Memeriksa solusi pencadangan berbasis cloud

Manfaat terkait aksesibilitas, otomatisasi, skalabilitas, dan keamanan mungkin menginspirasi Anda untuk menggunakan layanan cloud untuk pencadangan database dan pemulihan bencana. Jika ini adalah arah yang Anda pilih, pertimbangkan dengan cermat biaya jangka panjang, masalah keamanan data, dan potensi penguncian vendor. Temukan mitra yang dapat menjawab pertanyaan apa pun mengenai permasalahan tersebut — dan pastikan bahwa vendor lock-in tidak akan terjadi.

Meskipun layanan cloud pada awalnya bisa menghemat biaya, biaya jangka panjang dapat meningkat jika kebutuhan Anda akan penyimpanan data meningkat secara signifikan. Sehubungan dengan itu, mentransfer data dalam jumlah besar ke dalam atau ke luar cloud bisa jadi mahal, terutama bagi perusahaan dengan bandwidth terbatas. Selain itu, ada masalah kepatuhan dan keamanan data yang mungkin lebih penting bagi Anda dibandingkan bagi vendor. Meskipun vendor mungkin memberikan janji, tanggung jawab dan potensi dampaknya ada pada Anda.

Ada layanan cloud yang aman dan andal, tetapi bacalah rinciannya dan ajukan pertanyaan tersebut.

Jangan pernah berhemat pada langkah-langkah keamanan

Berikut beberapa praktik terbaik untuk membantu memastikan data tetap aman selama pencadangan dan pemulihan:

Enkripsi (yang juga bisa muncul di bagian Elemen Penting) adalah komponen yang harus dimiliki dalam strategi pencadangan dan pemulihan basis data. Menggunakan algoritme dan kunci untuk membuat data tidak dapat dibaca, enkripsi membantu melindungi informasi selama

pencadangan dan pemulihan. Bahkan jika file cadangan jatuh ke tangan yang salah, Anda aman. Enkripsi juga membantu Anda mematuhi peraturan perlindungan data yang ketat (GDPR, Sarbanes-Oxley Act, PCI DSS, CCPA, HIPAA, dll.).

Kontrol akses adalah proses mengelola siapa yang dapat mengakses database dan tindakan apa yang dapat mereka lakukan. Dalam pencadangan dan pemulihan, kontrol akses membantu mencegah akses tidak sah ke file cadangan sensitif dan mekanisme pemulihan basis data.

Otorisasi cadangan dan otorisasi pemulihan adalah proses menentukan apakah pengguna atau proses memiliki izin untuk melakukan tindakan tertentu. Otorisasi membantu memastikan bahwa hanya entitas resmi yang dapat memulai operasi pencadangan dan pemulihan.

Bersikaplah proaktif — atur pemantauan dan peringatan

Para penambang batu bara di masa lalu membawa burung kenari yang dikurung jauh ke dalam terowongan tambang. Jika ada karbon monoksida atau gas berbahaya lainnya, gas tersebut akan membunuh burung kenari, dan memberi isyarat kepada para penambang untuk segera meninggalkan terowongan.

Untungnya, dengan pencadangan database dan pemulihan bencana, memperhatikan tanda-tanda peringatan bisa menjadi jauh lebih ilmiah — dan sangat mudah. Daripada mengandalkan sayap dan doa, solusi pemantauan dan peringatan yang efektif dapat mengandalkan:

Ambang Batas: Ambang batas untuk berbagai metrik, seperti waktu penyelesaian pencadangan, jeda replikasi, dan ketersediaan sumber daya, ditentukan. Ketika ambang batas tersebut tercapai, peringatan akan dipicu.

Saluran pemberitahuan untuk peringatan waktu nyata: Dengan konfigurasi yang tepat, personel yang tepat akan segera dihubungi melalui email otomatis, teks, obrolan, dan saluran lain ketika ambang batas yang disebutkan sebelumnya tercapai. Pemberitahuan tersebut harus mencakup proses eskalasi di mana jalur dukungan atau orang yang berbeda akan dihubungi jika peringatan tersebut tidak diketahui atau diselesaikan dalam jangka waktu yang telah ditentukan.

Peningkatan penyimpanan otomatis: Harus ada otomatisasi — yang terkait dengan peringatan — di mana ruang penyimpanan ditingkatkan ketika mencapai ambang batas yang telah ditentukan. Ini akan membantu mencegah kegagalan pencadangan.

Log terperinci: Penting untuk menyimpan log semua aktivitas pemantauan dan peringatan. Kemudian, Anda memiliki informasi untuk menghasilkan laporan guna mengidentifikasi tren dan area perbaikan.

Dengan sistem pemantauan yang tepat , Anda dapat menghindari kerugian. Anda juga dapat menemukan masalah kinerja penting dengan lebih cepat, memahami akar penyebab insiden dengan lebih baik, dan memecahkan masalah tersebut dengan lebih efisien di masa mendatang.

Memulihkan dari skenario yang berbeda

Dalam perencanaan pencadangan database dan pemulihan bencana, langkah-langkah yang jelas untuk menangani kegagalan perangkat keras harus ada. Langkah-langkah tersebut harus mencakup:

Identifikasi: Langkah pertama dalam pemulihan dari kegagalan perangkat keras adalah mengidentifikasi komponen perangkat keras yang terpengaruh. Ini bisa berupa hard drive, server, atau bahkan seluruh pusat data. Alat dan sistem pemantauan dapat membantu mendeteksi kegagalan tersebut dan memicu peringatan otomatis yang dijelaskan di bagian sebelumnya.

Isolasi dan remediasi: Setelah kegagalan teridentifikasi, staf TI atau sistem otomatis harus bekerja untuk mengisolasi perangkat keras yang terkena dampak dan memulihkannya ke kondisi fungsional sesegera mungkin. Hal ini mungkin melibatkan menjadikan server offline, mengubah rute lalu lintas jaringan, atau mengganti drive disk yang gagal.

Restorasi: Setelah masalah perangkat keras teratasi, langkah selanjutnya adalah memulihkan layanan database. Hal ini melibatkan memulai ulang server basis data, memulihkan koneksi jaringan, dan memastikan bahwa sistem manajemen basis data beroperasi. Mekanisme failover otomatis dan penyeimbang beban yang disebutkan sebelumnya dapat membantu meminimalkan waktu henti selama fase ini.

Pemulihan data dari cadangan: Ini mungkin merupakan cadangan penuh atau kombinasi dari cadangan penuh dan tambahan. Gunakan perangkat lunak pencadangan untuk memulihkan database ke kondisi bersih pada saat pencadangan terakhir yang baik. Ini mungkin melibatkan penyalinan data dari penyimpanan cadangan ke server database produksi.

Mungkin kita harus menyebutnya Bagian Semua Hal yang Tidak Menyenangkan karena sekarang kita akan membahas pemulihan data setelah serangan siber. (Tentu saja kenari yang mati itu cukup gelap.)

Tapi sungguh, bagian ini adalah tentang bangkit kembali. Dengan persiapan yang tepat, serangan siber tidak harus menjadi lonceng kematian bagi sistem database dan bisnis Anda. Faktanya, Anda bisa mengalahkan serangan dan kembali lebih kuat. Jadi mari kita lihat perencanaan, kesiapan, dan pendekatan sistematis untuk meminimalkan downtime dan kehilangan data sekaligus memastikan keamanan dan integritas sistem dan informasi Anda. Berikut langkah-langkah penting dalam mengatasi serangan siber:

Persiapan: Lakukan pencadangan rutin untuk memastikan Anda memiliki salinan data terbaru dan bersih untuk dipulihkan. Simpan salinan cadangan di lokasi luar kantor atau terpencil, mungkin di cloud. Ini melindungi data dari kerusakan fisik atau gangguan jika terjadi serangan.

Dokumentasi: Pertahankan dokumentasi menyeluruh tentang konfigurasi database, skema, dan struktur data Anda. Dokumentasi ini akan sangat berharga selama proses restorasi.

Rencana respons: Kembangkan rencana respons insiden yang jelas yang menguraikan peran, tanggung jawab, dan langkah-langkah yang harus diambil jika terjadi serangan siber.

Deteksi dan isolasi: Segera setelah Anda mendeteksi serangan, segera identifikasi cakupan dan sifatnya. Tentukan database atau sistem mana yang terpengaruh. Kemudian, karantina atau putuskan sambungan sistem yang terkena dampak dari jaringan untuk mencegah penyebaran malware dan kerusakan data lebih lanjut.

Penilaian kerusakan: Evaluasi tingkat kehilangan atau kerusakan data. Penilaian ini akan membantu menentukan strategi restorasi yang tepat.

Identifikasi pelaku: Agar Anda dapat menambal kerentanan dan mencegah serangan di masa mendatang, tentukan bagaimana serangan itu terjadi.

Pemulihan data: Gunakan cadangan bersih terbaru Anda untuk memulihkan database yang terpengaruh. Pastikan cadangannya berasal dari waktu sebelum serangan terjadi. Dalam beberapa kasus ketika serangan telah membahayakan data, Anda mungkin harus melakukan pemulihan bertahap. Hal ini melibatkan penerapan pencadangan tambahan untuk membawa data sesuai standar.

Pembaruan dan audit keamanan: Segera menambal dan memperbarui sistem basis data dan perangkat lunak terkait untuk mengatasi kerentanan yang dieksploitasi. Untuk mencegah serangan di masa mendatang, terapkan sistem deteksi intrusi (IDS) dan kontrol akses.

Konsistensi dan integritas data

Anda harus menjaga keakuratan data sebelum, selama, dan setelah bencana atau serangan. Dengan melakukan hal ini, organisasi Anda dapat pulih dengan cepat dan andal. Selain replikasi, pemantauan dan peringatan, enkripsi, audit, dan aktivitas lain yang telah disebutkan, berikut beberapa praktik terbaik lainnya untuk menjaga keakuratan data dan mengatasi korupsi data:

Lakukan pengujian pencadangan rutin , penuh dan bertahap, untuk memverifikasi integritas data.

Lakukan verifikasi pencadangan otomatis untuk memeriksa integritas file cadangan dan memastikan tidak rusak.

Terapkan kontrol versi untuk pencadangan basis data , yang akan memberi Anda riwayat perubahan dan memungkinkan Anda memilih titik waktu tertentu jika diperlukan.

Selalu terapkan patch keamanan terbaru dan pembaruan pada sistem manajemen database dan perangkat lunak cadangan Anda untuk mencegah kerentanan.

Gunakan IDS dan alat manajemen informasi dan peristiwa keamanan (SIEM) untuk memantau aktivitas jaringan dan mendeteksi aktivitas mencurigakan.

Kembangkan dan perbarui secara berkala rencana pemulihan bencana komprehensif yang menguraikan peran, tanggung jawab, dan prosedur pemulihan data dalam berbagai skenario.

Pertimbangkan untuk meminta pakar dari luar menilai rencana perlindungan dan pemulihan data Anda dan memberikan rekomendasi untuk perbaikan.

Menskalakan proses pencadangan dan pemulihan

Data dalam jumlah besar dapat berada di database besar dan perusahaan. Meski tampak jelas, penting bagi Anda untuk tidak lengah, dan itu berarti memiliki kapasitas penyimpanan yang signifikan dan mekanisme transfer data yang efisien.

Perusahaan juga biasanya memerlukan pencadangan yang sering, mulai dari harian hingga setiap jam, bergantung pada RPO mereka. Proses pencadangan otomatis sangat penting dalam memastikan konsistensi data dan meminimalkan waktu henti. Teknik seperti pencadangan online dan pencadangan berbasis snapshot dapat membantu memastikan database tetap dapat diakses selama proses pencadangan.

Sehubungan dengan itu, mentransfer cadangan basis data yang besar melalui jaringan dapat membebani bandwidth yang tersedia. Oleh karena itu, perusahaan mungkin memerlukan koneksi khusus berkecepatan tinggi atau solusi pengoptimalan WAN untuk mengurangi kemacetan jaringan selama operasi pencadangan. Untuk mengurangi kebutuhan penyimpanan dan bandwidth, teknik kompresi dan deduplikasi sering diterapkan pada data cadangan. Ini melibatkan identifikasi data redundan dan hanya menyimpan blok unik.

Perencanaan kelangsungan usaha

Dalam real estate, mereka mengatakan “lokasi, lokasi, lokasi.” Dalam manajemen basis data, kami mengatakan: Proaktif. Proaktif. Proaktif. (Oke, tidak terlalu menarik dan mungkin kami tidak mengatakannya dengan kata-kata tersebut, namun kami memikirkannya.) Dan di sini, kami akan mengatakannya untuk keempat kalinya dalam kaitannya dengan perlindungan data dan menghindari downtime: Proaktif.

Setiap bisnis atau organisasi yang bergantung pada data (artinya hampir semuanya) harus proaktif jika ingin menjaga kelangsungan bisnis di tengah bencana, pemadaman listrik, serangan siber, atau peristiwa lain yang dapat mengancam proses data.

Kita membahas banyak hal tentang komponen, alat, dan praktik teknis terbaik, jadi di sini kita akan membahas bagian perencanaan sebenarnya yang harus disertakan dalam rencana kelangsungan bisnis. Karena replikasi dan redundansi adalah elemen penting dalam manajemen database, mari kita tetap menggunakan tema redundansi dan menyebut aktivitas proaktif berikut ini :

Penilaian risiko: Identifikasi potensi risiko dan ancaman yang dapat berdampak negatif pada sistem database Anda. Hal ini dapat mencakup bencana alam (angin topan, tornado, gempa bumi, banjir, badai salju, dll.), serangan siber (pelanggaran data, bug berbahaya, ransomware, dll.), kesalahan manusia, dan kegagalan perangkat keras.

Penilaian dampak: Evaluasi bagaimana setiap risiko atau ancaman yang teridentifikasi dapat berdampak negatif terhadap sistem database Anda.

Tujuan pemulihan: Untuk menentukan seberapa cepat organisasi Anda harus memulihkan sistem databasenya setelah terjadi gangguan (waktu henti maksimum yang diperbolehkan), Anda harus menetapkan RTO. Untuk menentukan jumlah maksimum kehilangan data yang dapat ditoleransi organisasi Anda, Anda harus menetapkan RPO. Ini akan menentukan seberapa sering Anda harus mencadangkan data Anda.

Rencana pemulihan bencana (DRP): Semua komponen di bagian ini merupakan bagian dari DRP, yang menguraikan langkah-langkah yang harus diambil jika terjadi bencana atau serangan siber. DRP harus mencakup peran dan tanggung jawab, prosedur komunikasi, dan prosedur pemulihan. Anda harus menguji DRP secara teratur melalui simulasi untuk memastikan DRP bekerja secara efektif.

Rencana komunikasi: Kembangkan rencana komunikasi untuk memberikan informasi kepada karyawan, pelanggan, dan pemangku kepentingan saat terjadi bencana atau serangan siber.

Rencana keuangan: Mengalokasikan sumber daya anggaran untuk pemulihan bencana dan inisiatif kelangsungan usaha untuk memastikan pendanaan yang memadai.

Alat tambahan dan keahlian dari luar: Rencana kesinambungan bisnis bukanlah sesuatu yang Anda buat begitu saja. Anda mungkin tidak memiliki alat dan keahlian staf untuk menyelesaikannya. Pertimbangkan perangkat lunak pencadangan yang siap digunakan dan apakah Anda dapat merancang, mengimplementasikan, dan memelihara rencana kelangsungan bisnis Anda sendiri atau tidak. Jika tidak, pertimbangkan bantuan dari luar , namun waspadalah terhadap lisensi kepemilikan dan jebakan vendor lock-in.

6.3 Manajemen Transaksi

Konsep Transaksi

Transaksi adalah sebuah unit eksekusi dari program yang mengakses dan memungkinkan update berbagai macam tipe data. Biasanya suatu transaksi diinisialisasikan oleh program user yang ditulis dalam bahasa pemrograman atau manipulasi data tingkat tinggi (sebagai contoh, SQL, C/C++), yang dibatasi oleh statement (pemanggilan fungsi) dalam bentuk begin transaction dan end transaction. Transaksi terdiri dari semua operasi yang dieksekusi diantara begin transaction dan end transaction.

Pada saat eksekusi transaksi, database bisa saja menjadi tidak konsisten. Namun pada saat transaksi sampai pada level commited, maka databasenya harus konsisten. Dua hal utama yang mungkin akan dihadapi pada saat melakukan transaksi :

- Terjadinya berbagai macam kegagalan, yang bisa disebabkan karena kegagalan hardware, system crash, dll
- Eksekusi konkuren (secara bersama) yang melibatkan banyak transaksi

Untuk memastikan integritas data tetap terjaga dan transaksi dapat berjalan dengan baik, maka sistem database harus menjaga properti – properti yang terdapat di dalam transaksi. Properti – properti di dalam transaksi ini dikenal dengan istilah Properti ACID (Atomicity, Consistency, Isolation, Durability). Properti ACID memastikan perilaku yang dapat diprediksi dan menguatkan peran transaksi sebagai konsep all or nothing yang didesain untuk mengurangi manajemen load ketika ada banyak variabel.

Properti ACID

Atomicity. Transaksi dilakukan sekali dan sifatnya atomik, artinya merupakan satu kesatuan tunggal yang tidak dapat dipisah – laksanakan pekerjaannya sampai selesai atau tidak sama sekali. Consistency. Jika basis data awalnya dalam keadaan konsisten maka pelaksanaan transaksi sendirinya juga harus meninggalkan basis data tetap dalam status konsisten. Isolation. Isolasi memastikan bahwa secara bersamaan eksekusi transaksi terisolasi dari yang lain. Durability. Begitu transaksi telah dilaksanakan (di-commit) maka perubahan yang dilakukan tidak akan hilang dan tetap terjaga (durable), sekalipun ada kegagalan sistem.

Transaksi mengakses data dengan operasi

Read(X), mentransfer data item X dari database ke local buffer yang dimiliki oleh transaksi yang mengeksekusi operasi pembacaan (read). Write(X), mentransfer data item X dari local buffer dari aksi transaksi yang mengeksekusi perintah penulisan kembali ke database (write)

Contoh implementasi transaksi, misalkan transaksi transfer uang sebesar $50 dari rekening A ke rekening B, maka transaksi tersebut dapat didefinisikan sebagai berikut :

1. read(A)

2. A := A – 50

3. write(A)

4. read(B)

5. B := B + 50

6. write(B)

Berdasarkan contoh, ditinjau dari kebutuhan properti ACID-nya, maka :

- Kebutuhan Konsistensi (Consistency Requierements) : Total jumlah rekening A + B harus tetap, tidak berubah setelah proses eksekusi transaksi.

- Kebutuhan Atomik (Atomicity Requeirements) : Jika transaksi gagal diantara tahap ke-3 dan tahap ke-6, maka sistem harus memastikan bahwa perubahan yang terjadi tidak disimpan ke database, atau akan terjadi inkonsistensi data. Dengan kata lain, selesaikan transaksi atau tidak sama sekali.

- Kebutuhan Durabilitas (Durability) : Pada saat eksekusi transaksi selesai dilaksanakan, dan user yang melakukan transaksi sudah diberitahu bahwa transfer yang dilakukannya sukses, maka harus dipastikan bahwa tidak ada kesalahan sistem yang akan terjadi yang menyebabkan hilangnya data yang berkaitan dengan proses transfer tersebut .

- Kebutuhan Isolasi (Isolation) : Jika diantara tahap ke-3 dan tahap ke-6 ada transaksi lain yang disisipkan, maka akan dapat menyebabkan inkonsistensi terhadap database (jumlah rekening A+B bisa jadi berkurang dari yang seharusnya). Untuk menghindari hal itu, maka transaksi bisa dieksekusi secara serial.

Tapi walaubagaimanapun, eksekusi banyak transaksi secara bersama – sama (konkuren) memiliki banyak keuntungan.

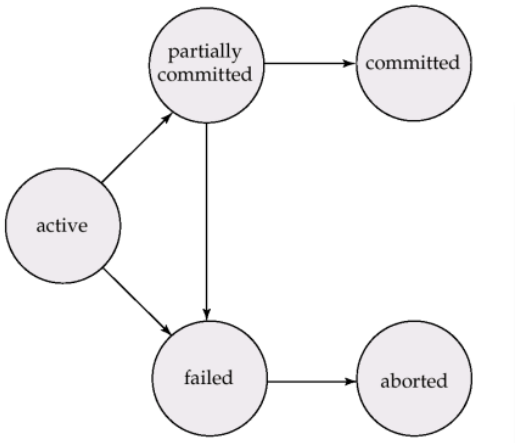
Supaya transaksi benar – benar sukses dipenuhi (successfully completion), maka sebuah transaksi harus berada di dalam salah satu state sebagai berikut :

- Active
- Partially committed
- Failed
- Aborted
- Commited

State tersebut dapat direpresentasikan dalam model transaksi abstrak yang sederhana

- Active merupakan initial state, transaksi tetap berada pada state ini pada saat proses eksekusi
- Partially Committed Setelah statement final telah dieksekusi
- Failed Setelah ditelusuri bahwa eksekusi normal tidak dapat diproses kembali
- Aborted Setelah transaksi di-rolled back dan database direstore ke kondisi awal sebelum transaksi dimulai
- Committed Setelah transaksi sukses dipenuhi

Diagram state yang menggambarkan proses transaksi :



Gambar 6. 1 Contoh Diagram State

Transaksi dimulai dalam keadaan state active. Pada saat menyelesaikan statement terakhirnya, transaksi masuk ke kondisi state partially commited. Dalam keadaan state tersebut, transaksi telah selesai dieksekusi, tapi masih mungkin untuk dibatalkan (aborted) hanya jika transaksi memasuki state aborted, karena output yang sesungguhnya masih berada di tempat penyimpanan sementara/main memory, dan karenanya kesalahan pada hardware dapat mempengaruhi kesuksesan dari penyelesaian transaksi Sistem Basis Data kemudian menuliskan informasi yang dibutuhkan (write) ke dalam disk, bahwa perubahan yang dilakukan oleh transaksi dapat dibuat kembali pada saat sistem restart jika terjadi kegagalan pada sistem. Pada saat informasi terakhir dituliskan, maka transaksi masuk ke kondisi state commited.

Sebuah transaksi akan memasuki kondisi state failed jika setelah sistem melakukan pemeriksaan, transaksi tidak dapat lagi diproses dengan eksekusi normal (misal karena kerusakan hardware atau kesalahan logika). Jika berada di dalam kondisi tersebut, maka transaksi harus di rolled back, dan kemudian selanjutnya memasuki kondisi state aborted (pembatalan transaksi). Pada titik ini, sistem memiliki dua opsi, ulangi transaksi (restart the transaction) dan hapus transaksi (kill the transaction)

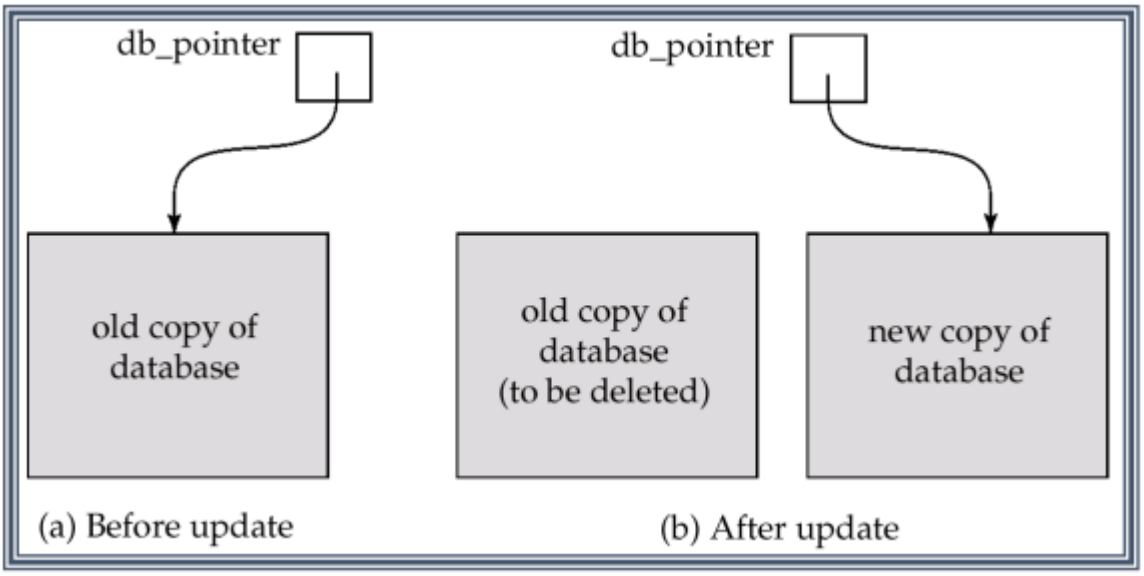
Opsi pada state aborted : Sistem dapat mengulang transaksi (restart the transaction), tapi hanya jika transaksi dibatalkan yang bisa terjadi karena adanya kerusakan software atau hardware yang tidak diciptakan melalui logika internal dari transaksinya. Sistem dapat menghapus transaksi (kill the transaction). Biasanya terjadi karena adanya kesalah logika internal yang dapat diperbaiki hanya dengan menulis kembali program aplikasinya, atau karena inputan yang tidak baik, atau karena data yang diinginkan tidak ada di database.

Implementasi Atomik (Atomicity) dan Durabilitas (Durability)

Komponen manajemen recovery dari sistem basis data dapat mendukung atomisitas dan durabilitas dengan berbagai macam skema. Salah satu skema yang dikenal adalah shadow copy, Shadow copy adalah salah satu skema yang digunakan untuk mendukung atomisitas dan durabilitas pada transaksi dengan membuat salinan / copy dari database yang ada.

Skema shadow-database :

- Mengasumsikan bahwa hanya satu transaksi yang aktif dalam waktu bersamaan (simpel tapi tidak efisien).
- Mengasumsikan bahwa database secara sederhana merupakan sebuah file di dalam disk.
- Sebuah pointer yang bernama db-pointer digunakan di dalam disk yang selalu mengarah ke salinan konsisten database tersebut
- Sebelum transaksi mengupdate database, salinan untuk database tersebut dibuat terlebih dahulu sepenuhnya
- Semua update dilakukan di salinan database yang baru, dan db_pointer akan mengarah ke salinan baru tersebut dengan catatan semua transaksi telah mencapai state partial committed dan semua update pagesnya telah diflush ke dalam disk.
- Salinan database tersebut kemudian menjadi database utama, dan database yang lama dapat dihapus.
- Jika transaksi gagal, maka db_pointer akan kembali mengarah ke database lama, dan salinan dari database yang telah dibuat tersebut dapat dihapus.



Gambar 6. 2 Contoh Skema shadow-database

Asumsi transaksi tidak gagal,Berguna untuk teks editor, tapi tidak efisien untuk database berukuran besar

Eksekusi Konkurensi

Pada eksekusi konkurensi, banyak transaksi dimungkinkan untuk diproses secara bersama – sama dalam suatu sistem. Memperbolehkan banyak transaksi untuk mengupdate data secara konkuren akan menyebabkan beberapa komplikasi dengan konsistensi data. Untuk memastikan bahwa konsistensi tetap terjaga dengan baik pada eksekusi konkuren, membutuhkan usaha yang lebih kuat. Akan lebih mudah jika transaksi berjalan secara serial.

Keuntungan konkurensi : Meningkatkan utilitas disk dan prosesor, Mengurangi rata – rata waktu respon transaksi. Dalam konkurensi dikenal dengan konsep penjadwalan (schedule) yang akan membantu mengidentifikasi eksekusi agar konsistensi datanya tetap terjaga. Sistem Basis Data harus mengontrol interaksi diantara transaksi konkuren supaya transaksi – transaksi tersebut tidak menghancurkan konsistensi basis datanya (data menjadi tidak konsisten).

Eksekusi Konkurensi – Schedule

Penjadwalan / Schedule merupakan urutan yang mengindikasikan urutan kronologis instruksi yang mana dari transaksi konkuren yang akan dieksekusi. Sebuah jadwal merupakan bagian dari suatu transaksi yang harus terdiri dari semua instruksi yang ada di dalamnya. Sebuah jadwal harus menjaga urutan instruksi yang muncul di setiap transaksi.

Misal T1 adalah transaksi transfer $50 dari A ke B, dan T2 adalah transfer 10% dari jumlah rekening A ke B. Penjadwalan berikut adalah pada transaksi serial, dimana T1 dulu yang diselesaikan baru diikuti oleh T2

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

Gambar 6. 3 Contoh Penjadwalan 1 di text book

Contoh penjadwalan selanjutnya, dengan transaksi yang sama, tetapi dalam bentuk transaksi konkurensi, tapi ekivalen dengan penjadwalan sebelumnya.

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

Gambar 6. 4 Contoh Penjadwalan 3 pada Text book

Pada contoh penjadwalan yang pertama dan yang kedua, jumlah rekening A + B sebelum dan sesudah transaksi sama, tapi tidak dengan contoh penjawalan konkurensi berikut :

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Gambar 6. 5 Contoh Penjadwalan 4 pada text book

Serializability

Sistem basis data harus dapat mengontrol eksekusi konkurensi dari suatu transaksi untuk memastikan database tetap terjaga konsistensinya. Asumsi dasar, bahwa setiap transaksi harus tetap menjaga konsistensi database. Eksekusi secara serial pada suatu transaksi dapat menjaga database tetap konsisten. Sebuah jadwal (yang mungkin konkuren), serializable jika setara dengan penjadwalan secara serial. Pada bagian ini ada dua bentuk ekivalensi penjadwalan (Schedule Equivalence) : Conflict dan View Serializability. Pada serializability kita abaikan operasi selain dari instruksi read dan write, serta diasumsikan bahwa transaksi dapat melakukan perhitungan terhadap data di dalam buffer lokal diantara instruksi read dan write.

Conflict Serializability

Instruksi I i dan I j masing – masing dari transaksi Ti dan Tj , konflik jika dan hanya jika ada beberapa item data yang sama (Q) diakses secara bersama – sama baik oleh I i dan I j , serta setidaknya salah satu dari instruksi melakukan operasi write (Q).

- Jika I i = read(Q) dan I j = read(Q), maka I i dan I j tidak konflik
- Jika I i = read(Q) dan I j = write(Q), maka I i dan I j konflik

- Jika I i = write(Q) dan I j = read(Q), maka I i dan I j konflik
- Jika I i = write(Q) dan I j = write(Q), maka I i dan I j konflik

Secara intuitif, konflik diantara I i dan I j memaksakan perintah logika temporal diantara keduanya. Jika I i dan I j merupakan instruksi dari transaksi berbeda dan tidak konflik, maka urutan instruksi I i dan I j dapat ditukar sehingga menghasilkan jadwal baru dengan hasil yang tetap sama.

Jika jadwal S dapat ditransformasikan menjadi jadwalan S'oleh serangkaian pertukaran instruksi yang non-konflik, maka bisa dikatakan S dan S' adalah conflict equivalent. Bisa dikatakan bahwa jadwal S adalah conflict serializable jika conflict equivalent terhadap penjadwalan serial. Contoh penjadwalan yang tidak conflict serializable:

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

Gambar 6. 6 Contoh Penjadwalan yang Tidak Conflict Serializable

Penjadwalan 3 di bawah bisa ditransformasikan ke penjadwalan 1, penjadwalan serial dimana T2 diikuti T1 , dengan serangkaian penukaran instruksi yang tidak konflik. Oleh karena itu penjadwalan di bawah ini conflict serializability:

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

Gambar 6. 7 Contoh conflict serializability

View Serializability

Misalkan S dan S' adalah dua jadwal dengan serangkaian transaksi yang sama. S dan S' adalah view equivalent. ϒ S dan S' adalah view equivalent jika memenuhi kondisi sebagai berikut :

- Untuk masing – masing data item Q , jika transaksi Ti membaca inisialisasi nilai pada Q di dalam jadwal S, maka transaksi Ti yang ada di jadwal S' juga harus membaca inisialisasi nilai pada Q.
- Untuk masing – masing data item Q, jika transaksi Ti mengeksekusi read(Q) di dalam jadwal S, dan jika nilai yang dihasilkan oleh operasi write(Q) dieksekusi oleh transaksi Tj , maka operasi read(Q) pada transaksi Ti yang harus ada di jadwal S' juga membaca nilai Q yang dihasilkan oleh operasi write(Q) yang sama pada transaksi Tj 3. Untuk masing – masing data item Q, transaksi yang sampai ke tahap akhir operasi write(Q) di dalam jadwal S

Seperti yang terlihat, view serializability juga dilihat berdasarkan pada operasi read dan write saja. Sebuah jadwal S adalah view seralizable , setara dengan penjadwalan serial. Setiap penjadwalan conflict serializable juga view serializable . Penjadwalan di bawah ini (penjadwalan 9 di text book) adalah penjadwalan yang view serializable tapi tidak conflict serializable:

T_3	T_4	T_6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

Gambar 6. 8 Contoh Penjadwalan yang View Serializable Tapi Tidak Conflict Serializable

Penjadwalan di bawah (penjadwalan 9 di text book) menghasilkan outcome yang sama sebagai jadwal serial (T1 , T2)

T_1	T_5
read(A)	
$A := A - 50$	
write(A)	
	read(B)
	$B := B - 10$
	write(B)
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$A := A + 10$
	write(A)

Gambar 6. 9 Contoh Penjadwalan 9 di Text Book

Recoverable Schedule – Jika transaksi Tj membaca sebuah item data yang sebelumnya ditulis oleh transaksi Ti , maka operasi commit Ti muncul sebelum operasi commit Tj . Jadwal berikut (jadwal 11 di text book) tidak recoverable jika T9 commit segera setelah read.

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

Gambar 6. 10 Contoh Jadwal 11 di Text Book

Jika T8 dibatalkan, maka T9 bisa jadi akan membaca state database inkonsisten. Maka dengan demikian database harus memastikan bahwa jadwal dapat dipulihkan jika terjadi sesuatu hal. Cascading Rollback – kesalahan pada satu transaksi yang yang dapat berpengaruh pada serangkaian transaksi lainnya sehingga keseluruhan transaksi akan di-rollback. Misalkan pada

penjadwalan berikut dimana belum ada transaksi yang dicommit, sehingga jika terjadi masalah bisa dipulihkan.

T_{10}	T_{11}	T_{12}
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)

Gambar 6. 11 Contoh Cascading Rollback

Jika T10 transaksinya fail, maka T11 dan T12 juga harus di rollback. Jika tidak demikian, maka akan menyebabkan banyaknya pekerjaan yang tidak akan terselesaikan. Cascadeless Schedule – cascading rollback tidak dapat terjadi, untuk setiap pasang transaksi Ti dan Tj dimana Tj membaca sebuah item data yang sebelumnya ditulis oleh Ti dan operasi commit pada Ti muncul sebelum operasi read pada Tj . Setiap cascadeless schedule juga dapat dipulihkan (recoverable).

Implementasi Isolasi

Jadwal harus conflict atau view serializable, dan recoverable, demi terjaganya konsistensi database dan juga sebaiknya cascadeless. Sebuah kebijakan dimana hanya satu transaksi yang dapat dieksekusi dalam satu waktu yang menghasilkan penjadwalan serial, tapi minim konkurensi. Beberapa skema hanya mengijinkan jadwal conflictserializable untuk di-generate, dimana yang skema lainnya ada yang memungkinkan jadwal viewserializable yang tidak conflict-serializable.

Definisi Transaksi di SQL

DML (Data Manipulation Language) harus menyertakan sebuah konstruksi untuk menspesifikasikan serangkaian aksi yang meliputi suatu transaksi. Di dalam SQL, transaksi dimulai secara implisit. Transaksi di SQL diakhiri oleh salah satu dari statement sebagai berikut:

- commit work - commit transaksi yang sedang berlangsung dan memulai transaksi yang baru

- rollback work – pembatalan terhadap transaksi yang sedang berlangsung

Level Konsistensi di SQL 92

Serializable – default, Repeatable read: hanya record yang dicommit yang dibaca, pembacaan secara berulang untuk record yang sama harus dapat menghasilkan nilai yang sama. Bagaimanapun, transaksi mungkin tidak serializable - bisa menemukan beberapa record yang ditambahkan oleh suatu transaksi tapi tidak menemukan record lainnya. Read commited: hanya record yang dicommit yang akan dibaca , tetap commit untuk pembacaan record secara berulang yang menghasilkan nilai yang berbeda. Read uncommitted: record yang tidak dicommit dibaca.

Testing Serializability

Pada saat mendesain skema kontrol konkurensi, kita harus tunjukan bahwa jadwal yang dibuat oleh skema tersebut adalah serializable. Terdapat metode simpel dan efisien untuk menentukan conflict serializability dari suatu jadwal. Misalkan sebuah jadwal S. Kita dapat membuat suatu grafik langsung yang diberi nama grafik preseden (presedence graph).

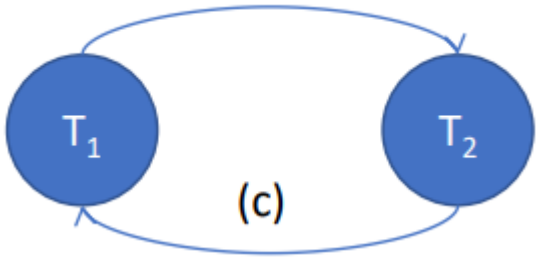
Grafik preseden terdiri dari sepasang G = (V,E), dimana V adalah serangkaian simpul dan E adalah serangkaian tepian / busur. Serangkaian simpul terdiri dari semua transaksi yang berperan serta di dalam penjadwalan. Serangkaian tepian / busur terdiri dari semua bentuk Ti -> Tj untuk masing – masing dari ketiga kondisi berikut :

- Ti eksekusi write(Q) sebelum Tj eksekusi read(Q)
- Ti eksekusi read(Q) sebelum Tj eksekusi write(Q)
- Ti eksekusi write(Q) sebelum Tj eksekusi write(Q)

Jika bentuk Ti -> Tj ada di dalam grafik preseden, maka di setiap jadwal S' serial yang ekivalen ke jadwal S, Ti harus muncul sebelum Tj.

Gambar 6. 12 Contoh Grafik Preseden Penjadwalan 1

Sebagai contoh, grafik preseden untuk penjadwalan 1 digambarkan di (a), terdiri dari bentuk dasar T1 -> T2 , dimana semua instruksi T1 dieksekusi sebelum instruksi pertama pada T2 dieksekusi. Begitu juga sebaliknya untuk contoh yang digambarkan di (b).



Gambar 6. 13 Contoh Grafik Preseden Penjadwalan 4

Grafik preseden untuk jadwal 4, terdiri dari bentuk T1 - > T2 , karena T1 mengeksekusi read(A) sebelum T2 mengeksekusi write(A). Grafik ini jg terdiri dari bentuk T2 -> T1 , karena T2 mengeksekusi read(B) sebelum T1 eksekusi write(B). Jika grafik preseden untuk S membentuk suatu lingkaran, maka jadwal S tidak conflict serializable. Jika sebaliknya, maka jadwal S conflict serializable.

6.4 Pemantauan dan Tuning Kinerja

Pemantauan dan tuning kinerja adalah aspek penting dalam operasi sistem basis data yang efektif dan efisien. Proses ini melibatkan pengawasan dan penyesuaian yang berkelanjutan terhadap kinerja basis data untuk memastikan bahwa sistem beroperasi dengan optimal. Berikut adalah panduan yang mendalam tentang pemantauan dan tuning kinerja dalam operasi sistem basis data:

- **Pemantauan Kinerja**: Pemantauan kinerja basis data melibatkan pengawasan yang terus-menerus terhadap berbagai metrik kinerja, termasuk waktu tanggap, throughput, penggunaan CPU, penggunaan memori, dan utilitas penyimpanan. Ini memungkinkan administrator basis data untuk mendeteksi masalah kinerja sejak dini dan mengambil tindakan korektif dengan cepat.
- **Identifikasi dan Analisis Bottleneck**: Salah satu langkah penting dalam pemantauan kinerja adalah mengidentifikasi bottleneck atau titik lemah dalam sistem. Ini bisa terjadi di berbagai tingkat, mulai dari infrastruktur fisik hingga konfigurasi basis data dan logika aplikasi. Analisis mendalam diperlukan untuk memahami penyebabnya dan merancang solusi yang tepat.
- **Tuning Konfigurasi Basis Data**: Tuning konfigurasi basis data melibatkan penyesuaian parameter dan pengaturan basis data untuk meningkatkan kinerja dan efisiensi. Ini termasuk pengaturan parameter seperti ukuran buffer cache, konfigurasi pengindeksan, strategi caching, dan tuning query untuk mengoptimalkan eksekusi query.

- **Pengaturan Struktur Data**: Struktur data yang efisien dan terorganisasi dengan baik merupakan kunci untuk kinerja yang optimal. Pemantauan dan tuning struktur data melibatkan pengoptimalan indeks, partisi tabel, dan pemilihan skema penyimpanan yang tepat untuk meminimalkan overhead dan meningkatkan akses data.

- **Pemantauan Pemakaian Sumber Daya**: Selain memantau kinerja basis data, penting juga untuk memantau pemakaian sumber daya seperti CPU, memori, dan penyimpanan. Hal ini memungkinkan identifikasi dan penanganan masalah yang berkaitan dengan sumber daya yang terbatas atau diperebutkan di lingkungan multi-tenant atau cloud.

- **Pemantauan Anomali dan Kecurangan**: Pemantauan kinerja juga harus mencakup deteksi anomali dan aktivitas yang mencurigakan yang dapat mengindikasikan serangan keamanan atau perilaku tidak biasa. Ini melibatkan penggunaan teknik pemantauan log dan analisis perilaku untuk mengidentifikasi potensi ancaman dan respon cepat terhadap insiden keamanan.
- **Automatisasi dan Pengkodean**: Untuk meningkatkan efisiensi dan responsifitas, pemantauan kinerja dapat diotomatisasi sebanyak mungkin menggunakan alat pemantauan dan manajemen basis data otomatis. Ini termasuk pengkodean otomatis untuk mengelola konfigurasi dan tindakan penyesuaian berdasarkan aturan dan kebijakan yang ditentukan sebelumnya.

- **Evaluasi dan Perbaikan Berkelanjutan**: Proses pemantauan dan tuning kinerja harus menjadi bagian dari siklus hidup pengembangan dan operasi sistem basis data. Evaluasi terus-menerus tentang efektivitas strategi tuning yang diterapkan, dan identifikasi area untuk perbaikan berkelanjutan adalah kunci untuk memastikan kinerja yang optimal dalam jangka panjang.

Dengan menerapkan praktik pemantauan dan tuning kinerja yang tepat, organisasi dapat meningkatkan kinerja, keandalan, dan skalabilitas sistem basis data mereka. Ini tidak hanya meningkatkan pengalaman pengguna akhir, tetapi juga memungkinkan organisasi untuk memenuhi kebutuhan bisnis yang semakin kompleks dan dinamis di era digital saat ini.

Bab 7 Keamanan Sistem Basis Data

7.1 Ancaman Keamanan Data

Keamanan basis data mencakup berbagai tindakan yang digunakan untuk mengamankan sistem manajemen basis data dari serangan dunia maya yang berbahaya dan penggunaan yang tidak sah. Program keamanan basis data dirancang untuk melindungi tidak hanya data di dalam basis data, namun juga sistem manajemen data itu sendiri, dan setiap aplikasi yang mengaksesnya, dari penyalahgunaan, kerusakan, dan intrusi.

Keamanan basis data mencakup alat, proses, dan metodologi yang membangun keamanan di dalam lingkungan basis data.

Ancaman Keamanan Basis Data

Banyak kerentanan perangkat lunak , kesalahan konfigurasi, atau pola penyalahgunaan atau kecerobohan dapat mengakibatkan pelanggaran. Berikut adalah sejumlah penyebab dan jenis ancaman siber terhadap keamanan basis data yang paling diketahui .

Ancaman Orang Dalam

Ancaman orang dalam adalah risiko keamanan yang berasal dari salah satu dari tiga sumber berikut, yang masing-masing memiliki cara istimewa untuk masuk ke database:

- Orang dalam yang jahat dengan niat buruk
- Orang yang lalai dalam organisasi yang membuat database diserang melalui tindakan ceroboh
- Orang luar yang memperoleh kredensial melalui rekayasa sosial atau metode lain, atau memperoleh akses ke kredensial database

Ancaman orang dalam adalah salah satu penyebab paling umum dari pelanggaran keamanan basis data dan sering kali terjadi karena banyak karyawan telah diberikan akses pengguna yang istimewa.

Kesalahan manusia

Kata sandi yang lemah, pembagian kata sandi, penghapusan atau kerusakan data yang tidak disengaja, dan perilaku pengguna yang tidak diinginkan lainnya masih menjadi penyebab hampir separuh pelanggaran data yang dilaporkan.

Eksploitasi Kerentanan Perangkat Lunak Basis Data

Penyerang terus-menerus berupaya mengisolasi dan menargetkan kerentanan dalam perangkat lunak, dan perangkat lunak manajemen basis data adalah target yang sangat berharga. Kerentanan baru ditemukan setiap hari, dan semua platform manajemen basis data sumber terbuka dan vendor perangkat lunak basis data komersial mengeluarkan patch keamanan secara berkala. Namun, jika Anda tidak menggunakan patch ini dengan cepat, database Anda mungkin terkena serangan.

Bahkan jika Anda menerapkan patch tepat waktu, selalu ada risiko serangan zero-day , ketika penyerang menemukan kerentanan, namun kerentanan tersebut belum ditemukan dan ditambal oleh vendor database.

Serangan Injeksi SQL/NoSQL

Ancaman khusus basis data melibatkan penggunaan string serangan non-SQL dan SQL yang sewenang-wenang ke dalam kueri basis data. Biasanya, ini adalah kueri yang dibuat sebagai perpanjangan formulir aplikasi web , atau diterima melalui permintaan HTTP. Sistem basis data apa pun rentan terhadap serangan ini, jika pengembang tidak mematuhi praktik pengkodean yang aman, dan jika organisasi tidak melakukan pengujian kerentanan secara rutin.

Serangan Buffer Overflow

Buffer overflow terjadi ketika suatu proses mencoba menulis sejumlah besar data ke dalam blok memori dengan panjang tetap, lebih dari yang diizinkan untuk ditampung. Penyerang mungkin menggunakan kelebihan data, yang disimpan di alamat memori yang berdekatan, sebagai titik awal untuk melancarkan serangan.

Serangan Penolakan Layanan (DoS/DDoS).

Dalam serangan penolakan layanan (DoS), penjahat dunia maya menguasai layanan target—dalam hal ini server database—menggunakan sejumlah besar permintaan palsu. Hasilnya adalah server tidak dapat menjalankan permintaan asli dari pengguna sebenarnya, dan sering kali crash atau menjadi tidak stabil.

Dalam serangan penolakan layanan terdistribusi (DDoS), lalu lintas palsu dihasilkan oleh sejumlah besar komputer, yang berpartisipasi dalam botnet yang dikendalikan oleh penyerang. Hal ini menghasilkan volume lalu lintas yang sangat besar, yang sulit dihentikan tanpa arsitektur pertahanan yang sangat terukur. Layanan perlindungan DDoS berbasis cloud dapat ditingkatkan secara dinamis untuk mengatasi serangan DDoS yang sangat besar .

Perusak Perangkat Lunak

Malware adalah perangkat lunak yang dibuat untuk memanfaatkan kerentanan atau menyebabkan kerusakan pada database. Malware dapat masuk melalui perangkat titik akhir mana pun yang terhubung ke jaringan database. Perlindungan malware penting pada endpoint mana pun, khususnya pada server database, karena nilai dan sensitivitasnya yang tinggi.

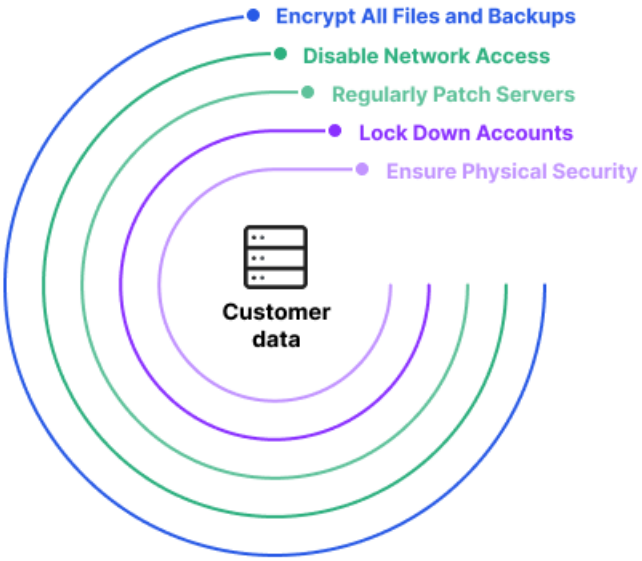
Lingkungan TI yang Berkembang

Lingkungan TI yang terus berkembang membuat database lebih rentan terhadap ancaman. Berikut adalah tren yang dapat menyebabkan jenis serangan baru terhadap database, atau mungkin memerlukan tindakan defensif baru:

- Pertumbuhan volume data —penyimpanan, pengambilan data, dan pemrosesan tumbuh secara eksponensial di hampir semua organisasi. Praktik atau alat keamanan data apa pun harus sangat terukur untuk memenuhi kebutuhan jangka panjang dan dalam waktu dekat.
- Infrastruktur terdistribusi— kompleksitas lingkungan jaringan semakin meningkat, terutama ketika bisnis mentransfer beban kerja ke arsitektur hybrid cloud atau multi-cloud, sehingga penerapan, pengelolaan, dan pilihan solusi keamanan menjadi lebih sulit.
- Persyaratan peraturan yang semakin ketat — lanskap kepatuhan terhadap peraturan di seluruh dunia semakin kompleks, sehingga memenuhi semua mandat menjadi semakin sulit.
- Kekurangan keterampilan keamanan siber — terdapat kekurangan profesional keamanan siber yang terampil secara global , dan banyak organisasi mengalami kesulitan untuk mengisi peran keamanan. Hal ini dapat mempersulit pertahanan infrastruktur penting, termasuk basis data.

Bagaimana Anda Mengamankan Server Basis Data Anda

Server basis data adalah mesin fisik atau virtual yang menjalankan basis data. Mengamankan server database, juga dikenal sebagai "pengerasan", adalah proses yang mencakup keamanan fisik, keamanan jaringan, dan konfigurasi sistem operasi yang aman.



Gambar 7. 1 Contoh Custumer Data

Pastikan Keamanan Basis Data Fisik

Jangan berbagi server untuk aplikasi web dan aplikasi database, jika database Anda berisi data sensitif . Meskipun mungkin lebih murah dan mudah untuk meng-host situs dan database Anda bersama-sama di penyedia hosting, Anda menyerahkan keamanan data Anda ke tangan orang lain.

Jika Anda mengandalkan layanan hosting web untuk mengelola database Anda, Anda harus memastikan bahwa layanan tersebut adalah perusahaan dengan rekam jejak keamanan yang kuat. Yang terbaik adalah menghindari layanan hosting gratis karena kemungkinan kurangnya keamanan.

Jika Anda mengelola database di pusat data lokal, perlu diingat bahwa pusat data Anda juga rentan terhadap serangan dari pihak luar atau ancaman orang dalam. Pastikan Anda memiliki langkah-langkah keamanan fisik, termasuk kunci, kamera, dan petugas keamanan di fasilitas fisik Anda. Akses apa pun ke server fisik harus dicatat dan hanya diberikan kepada individu yang berwenang.

Selain itu, jangan tinggalkan cadangan database di lokasi yang dapat diakses publik, seperti partisi sementara, folder web, atau keranjang penyimpanan cloud yang tidak aman.

Kunci Akun dan Hak Istimewa

Mari kita pertimbangkan server database Oracle. Setelah database diinstal, asisten konfigurasi database Oracle (DBCA) secara otomatis kedaluwarsa dan mengunci sebagian besar akun pengguna database default.

Jika Anda menginstal database Oracle secara manual, hal ini tidak akan terjadi dan akun dengan hak istimewa default tidak akan kedaluwarsa atau dikunci. Kata sandi mereka tetap sama dengan nama pengguna mereka, secara default. Penyerang akan mencoba menggunakan kredensial ini terlebih dahulu untuk terhubung ke database.

Penting untuk memastikan bahwa setiap akun istimewa di server database dikonfigurasi dengan kata sandi yang kuat dan unik. Jika akun tidak diperlukan, akun tersebut harus kedaluwarsa dan dikunci.

Untuk akun lainnya, akses harus dibatasi hingga jumlah minimum absolut yang diperlukan. Setiap akun hanya boleh memiliki akses ke tabel dan operasi (misalnya, SELECT atau INSERT) yang diperlukan oleh pengguna. Hindari membuat akun pengguna dengan akses ke setiap tabel di database.

Patch server Database secara rutin

Pastikan tambalan tetap terkini. Manajemen patch basis data yang efektif adalah praktik keamanan yang penting karena penyerang secara aktif mencari kelemahan keamanan baru dalam basis data, dan virus serta malware baru muncul setiap hari.

Penyebaran versi paket layanan basis data terkini, perbaikan keamanan penting, dan pembaruan kumulatif secara tepat waktu akan meningkatkan stabilitas kinerja basis data.

Nonaktifkan Akses Jaringan Publik

Organisasi menyimpan aplikasi mereka dalam database. Dalam sebagian besar skenario dunia nyata, pengguna akhir tidak memerlukan akses langsung ke database. Oleh karena itu, Anda harus memblokir semua akses jaringan publik ke server database kecuali Anda adalah penyedia hosting. Idealnya, organisasi harus menyiapkan server gateway (terowongan VPN atau SSH) untuk administrator jarak jauh.

Enkripsi Semua File dan Cadangan

Terlepas dari seberapa kuat pertahanan Anda, selalu ada kemungkinan bahwa peretas dapat menyusup ke sistem Anda. Namun, penyerang bukanlah satu-satunya ancaman terhadap keamanan database Anda. Karyawan Anda juga dapat menimbulkan risiko bagi bisnis Anda. Selalu ada kemungkinan bahwa orang dalam yang jahat atau ceroboh akan mendapatkan akses ke file yang mereka tidak punya izin untuk mengaksesnya.

Mengenkripsi data Anda membuatnya tidak dapat dibaca oleh penyerang dan karyawan. Tanpa kunci enkripsi, mereka tidak dapat mengaksesnya, hal ini memberikan garis pertahanan terakhir terhadap intrusi yang tidak diinginkan . Enkripsi semua file aplikasi, file data, dan cadangan penting sehingga pengguna yang tidak berwenang tidak dapat membaca data penting Anda.

Praktik Terbaik Keamanan Basis Data

Berikut beberapa praktik terbaik yang dapat Anda gunakan untuk meningkatkan keamanan database sensitif.

Kelola Kata Sandi dan Akses Pengguna secara Aktif

Jika Anda memiliki organisasi besar, Anda harus mempertimbangkan untuk mengotomatiskan manajemen akses melalui manajemen kata sandi atau perangkat lunak manajemen akses. Ini akan memberikan pengguna yang diizinkan kata sandi jangka pendek dengan hak yang mereka perlukan setiap kali mereka perlu mendapatkan akses ke database.

Itu juga melacak aktivitas yang diselesaikan selama jangka waktu tersebut dan menghentikan administrator membagikan kata sandi. Meskipun administrator mungkin merasa bahwa berbagi kata sandi itu mudah, namun hal ini membuat akuntabilitas dan keamanan database yang efektif hampir mustahil dilakukan.

Selain itu, langkah-langkah keamanan berikut direkomendasikan:

- Kata sandi yang kuat harus ditegakkan
- Hash kata sandi harus diasinkan dan disimpan terenkripsi
- Akun harus dikunci setelah beberapa kali upaya login
- Akun harus ditinjau secara berkala dan dinonaktifkan jika staf berpindah ke peran yang berbeda, meninggalkan perusahaan, atau tidak lagi memerlukan tingkat akses yang sama

Uji Keamanan Basis Data Anda

Setelah Anda menyiapkan infrastruktur keamanan basis data, Anda harus mengujinya terhadap ancaman nyata. Mengaudit atau melakukan uji penetrasi terhadap database Anda sendiri akan membantu Anda memahami pola pikir penjahat dunia maya dan mengisolasi kerentanan apa pun yang mungkin Anda abaikan.

Untuk memastikan pengujian ini komprehensif, libatkan peretas etis atau layanan pengujian penetrasi yang diakui dalam pengujian keamanan Anda. Penguji penetrasi memberikan laporan ekstensif yang mencantumkan kerentanan database, dan penting untuk segera menyelidiki dan memulihkan kerentanan ini. Jalankan uji penetrasi pada sistem database penting setidaknya sekali setahun.

Gunakan Pemantauan Basis Data Waktu Nyata

Memindai database Anda secara terus-menerus untuk mencari upaya pelanggaran akan meningkatkan keamanan Anda dan memungkinkan Anda bereaksi dengan cepat terhadap kemungkinan serangan.

Secara khusus, Pemantauan Integritas File (FIM) dapat membantu Anda mencatat semua tindakan yang dilakukan di server database dan memperingatkan Anda tentang potensi pelanggaran. Ketika FIM mendeteksi perubahan pada file database penting, pastikan tim keamanan diperingatkan dan mampu menyelidiki dan merespons ancaman tersebut.

Gunakan Aplikasi Web dan Firewall Basis Data

Anda harus menggunakan firewall untuk melindungi server database Anda dari ancaman keamanan database. Secara default, firewall tidak mengizinkan akses ke lalu lintas. Ini juga perlu menghentikan database Anda untuk memulai koneksi keluar kecuali ada alasan khusus untuk melakukannya.

Selain melindungi database dengan firewall, Anda harus menerapkan firewall aplikasi web (WAF). Hal ini karena serangan yang ditujukan pada aplikasi web, termasuk injeksi SQL , dapat digunakan untuk mendapatkan akses terlarang ke database Anda.

Firewall database tidak akan menghentikan sebagian besar serangan aplikasi web, karena firewall tradisional beroperasi pada lapisan jaringan, sedangkan lapisan aplikasi web beroperasi pada lapisan aplikasi (lapisan 7 model OSI). WAF beroperasi pada lapisan 7 dan mampu mendeteksi lalu lintas aplikasi web berbahaya, seperti serangan injeksi SQL, dan memblokirnya sebelum dapat membahayakan database Anda.

Keamanan Basis Data Imperva

Imperva menyediakan Firewall Aplikasi Web terdepan di industri , yang dapat mencegah serangan aplikasi web yang memengaruhi database, termasuk injeksi SQL. Kami juga menyediakan perlindungan integritas file (FIM) dan teknologi keamanan file, melindungi file sensitif dari penjahat dunia maya dan orang dalam yang jahat.

Selain itu, Imperva melindungi semua penyimpanan data berbasis cloud untuk memastikan kepatuhan dan menjaga ketangkasan dan manfaat biaya yang Anda dapatkan dari investasi cloud Anda:

Keamanan Data Cloud – Sederhanakan pengamanan database cloud Anda untuk mengikuti dan mengikuti perkembangan DevOps . Solusi Imperva memungkinkan pengguna layanan yang dikelola cloud untuk mendapatkan visibilitas dan kontrol data cloud dengan cepat.

Keamanan Basis Data – Imperva memberikan analisis, perlindungan, dan respons di seluruh aset data Anda, baik di lokasi maupun di cloud – memberi Anda visibilitas risiko untuk mencegah pelanggaran data dan menghindari insiden kepatuhan. Integrasikan dengan database apa pun untuk mendapatkan visibilitas instan, menerapkan kebijakan universal, dan mempercepat waktu untuk menilai.

Analisis Risiko Data – Otomatiskan deteksi perilaku akses data yang tidak patuh, berisiko, atau berbahaya di seluruh database Anda di seluruh perusahaan untuk mempercepat remediasi.

7.2 Strategi Perlindungan Data

Mengapa Memiliki Strategi Perlindungan Data Penting

Untuk mempercayai pelanggan dalam penanganan data sensitif, organisasi perlu menunjukkan pengembangan transparansi, integritas, dan keamanan di semua tahap penyimpanan dan pengumpulan data pribadi. Komitmen strategi untuk melindungi privasi pelanggan sangat penting untuk membangun kepercayaan. Inilah alasannya mengapa perlindungan data harus diterapkan sebagai bagian inti dari strategi bisnis.

Namun, perlindungan data dapat menjadi tantangan yang sulit bagi sebagian besar bisnis saat ini, bahkan perusahaan besar pun mengalami kesulitan. Pelanggaran data telah meningkat secara dramatis dalam jumlah dan ukuran dalam beberapa tahun terakhir, sehingga berdampak besar pada reputasi dan keuangan perusahaan-perusahaan yang terkena dampaknya.

Studi terbaru menunjukkan bahwa rata-rata kerugian akibat pelanggaran mencapai $4 juta, dan terus meningkat secara konsisten selama beberapa tahun. Menurut Risk Based Security ,

pada paruh pertama tahun 2021 saja terdapat 1.767 laporan pelanggaran di seluruh dunia, yang menghasilkan 18,8 miliar catatan.

Ketika kejahatan dunia maya menjadi semakin canggih, badan pengatur menciptakan undang-undang perlindungan data yang lebih ketat yang mempengaruhi perusahaan dan individu di seluruh dunia. Banyaknya peraturan ini mengharuskan perusahaan untuk menetapkan aturan dan langkah yang jelas untuk melindungi informasi pribadi. Mematuhi peraturan perlindungan data ini memerlukan perubahan signifikan pada proses dan perubahan budaya perusahaan—perubahan yang sulit dicapai tanpa adanya strategi yang solid.

10 Komponen Strategi Perlindungan Data yang Berhasil

Strategi perlindungan data yang berhasil biasanya mencakup komponen-komponen berikut.

1. Manajemen Siklus Hidup Data

Manajemen siklus hidup data adalah kerangka kerja yang menstandarkan proses data dalam organisasi, mulai dari pembuatan data, penyimpanan, pengarsipan, dan hingga penghapusan akhir. Manajemen siklus hidup data adalah komponen inti dari strategi perlindungan data yang solid.

2. Data Manajemen Risiko

Untuk melindungi data dengan benar, organisasi harus terlebih dahulu mengidentifikasi dan menilai semua risiko dan ancaman yang mungkin mempengaruhi data. Strategi perlindungan data perlu mempertimbangkan risiko dan ancaman ini dan mencakup langkah-langkah yang dirancang untuk meminimalkan dan memitigasi risiko ini.

3. Pencadangan dan Pemulihan Data

Tindakan pencadangan dan pemulihan merupakan komponen penting dalam strategi perlindungan data. Setelah data dibuat, data tersebut harus dicadangkan untuk memastikan dapat terjamin jika terjadi kegagalan.

Strategi perlindungan data harus menentukan jenis data apa yang harus dicadangkan, bagaimana data harus terjadi ketika terjadi bencana, dan media penyimpanan apa yang harus digunakan. Semua tindakan ini juga harus dimasukkan ke dalam inisiatif berkelanjutan bisnis dan pemulihan bencana (BCDR).

4. Kontrol Manajemen Akses Data

Kontrol akses merupakan aspek penting yang harus diterapkan dan dipelihara dengan baik. Kontrol ini memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data dan sistem perusahaan, dan dapat mencegah akses, penggunaan, atau transfer data yang tidak sah. Aspek ini sering diperiksa dan dinilai oleh auditor eksternal.

5. Manajemen Penyimpanan Data

Manajemen penyimpanan data mencakup tugas yang terkait dengan transmisi data produksi secara aman ke penyimpanan data, baik di lingkungan lokal atau di lingkungan cloud eksternal. Ini mungkin penyimpanan data yang ditujukan untuk akses yang sering dan berkinerja tinggi, atau penyimpanan arsip yang ditujukan untuk akses yang jarang.

6. Data Pencegahan Pelanggaran

Tindakan pencegahan pelanggaran data diterapkan dengan tujuan mencegah akses tidak sah terhadap data. Tujuannya adalah untuk mencegah pelaku jahat eksternal atau ancaman internal mendapatkan akses tidak sah ke data dan sistem.

Langkah-langkah keamanan siber diterapkan dengan tujuan mencegah serangan terhadap jaringan internal, perimeter jaringan, data dalam transit, dan data dalam penyimpanan. Biasanya, langkah-langkah ini mencakup enkripsi data, standar perangkat lunak antivirus, perlindungan terhadap ransomware, perangkat keras dan perangkat lunak keamanan perimeter, dan perangkat lunak manajemen akses.

7. Kerahasiaan, Integritas dan Ketersediaan

Kerahasiaan, integritas, dan ketersediaan (CIA), juga dikenal sebagai triad CIA, merupakan komponen kunci yang harus dijaga untuk memastikan perlindungan data. Strategi perlindungan data harus mendefinisikan standar CIA dan langkah-langkah untuk mempertahankannya.

8. Data Kebijakan dan Prosedur Perlindungan

Kebijakan menentukan aktivitas perlindungan data apa yang digunakan organisasi dan prosedur menentukan bagaimana aktivitas ini diterapkan. Kebijakan dan prosedur merupakan komponen penting dari program pengelolaan data, dan harus didokumentasikan dengan baik, karena biasanya diperiksa selama proses audit.

9. Standar dan Kepatuhan Terhadap Peraturan

Standar industri membantu organisasi mempertahankan perlindungan data yang mampu dan terkini. Badan mematuhi peraturan menentukan langkah-langkah yang dirancang untuk melindungi data, yang mana organisasi diwajibkan oleh hukum untuk mematuhinya. Setiap peraturan yang relevan dengan bisnis, industri, dan lokasi tertentu.

Mungkin peraturan yang paling umum dikenal adalah Peraturan Perlindungan Data Umum (GDPR) Uni Eropa (UE), yang melindungi data pribadi setiap warga negara UE. Namun, ada banyak peraturan dan standar industri lain yang harus dipatuhi oleh organisasi. Strategi perlindungan data harus mempertimbangkan semua peraturan yang relevan dan menentukan cara organisasi menjaga kepatuhan.

10. Pemantauan dan Peninjauan

Proses pemantauan dan peninjauan membantu organisasi mendapatkan visibilitas terhadap aktivitas data, risiko, dan kontrol, membantu meningkatkan perlindungan dan menanggapi ancaman dan anomali. Pemantauan dan peninjauan mungkin juga diperlukan untuk memenuhi persyaratan kepatuhan.

Pemantauan berkelanjutan memberikan visibilitas ke seluruh aspek siklus kehidupan data, termasuk pembuatan, penyimpanan, transmisi, pengarsipan, dan pemusnahan data. Aktivitas ini memberikan bukti penting bagi auditor internal dan eksternal yang memeriksa pengendalian yang ditetapkan untuk perlindungan dan pengelolaan data.

Praktik Terbaik Strategi Perlindungan Data

Strategi perlindungan data Anda harus lebih dari sekedar mematuhi peraturan seperti GDPR dan CCPA. Strategi ini menjalankan operasi bisnis Anda dan membantu mengamankan infrastruktur dan data Anda, menggunakan praktik berikut.

Pastikan Pemangku Kepentingan Utama selalu mengetahui informasinya

Pastikan pemangku kepentingan utama memahami strategi perlindungan data Anda dan menyetujuinya. Hal ini akan membantu memastikan bahwa karyawan mematuhi strategi dan menerapkan perlindungan data di seluruh organisasi, tidak hanya menyerahkannya ke TI.

Pantau Semua Data yang Tersedia

Buat inventarisasi data yang mencakup semua informasi yang disimpan atau diproses organisasi Anda. oleh organisasi Anda. Anda harus memahami data Anda untuk melindunginya—perhatian jenis data yang dikumpulkan, lokasi penyimpanannya, kebijakan penggunaan dan pembagiannya. Hal ini memungkinkan Anda memetakan sistem data Anda dan memfasilitasi pengelolaan.

Melakukan Analisis Risiko

Beberapa peraturan mengharuskan perusahaan untuk secara proaktif mengidentifikasi risiko dan mengambil tindakan untuk memitigasinya. Penilaian risiko sangat penting untuk menjadikan organisasi Anda akuntabel dan memungkinkan Anda mengidentifikasi potensi ancaman atau kekurangan. Infrastruktur bisnis Anda adalah web yang kompleks, dengan banyak jalur untuk mentransfer data—setiap jalur mempunyai potensi risiko, dan Anda harus melindungi data bahkan saat digunakan oleh pihak ketiga.

Lakukan analisis risiko untuk mengidentifikasi risiko individu di seluruh jaringan Anda. Ini akan membantu menginformasikan kebijakan perlindungan data Anda.

Perlindungan Data dengan Cloudian HyperStore

Perlindungan data memerlukan teknologi penyimpanan yang kuat. Peralatan penyimpanan Cloudian mudah diterapkan dan digunakan, memungkinkan Anda menyimpan data berukuran Petabyte dan mengaksesnya secara instan. Cloudian mendukung pencadangan dan pemulihan berkecepatan tinggi dengan transfer data paralel (18 TB per jam penulisan dengan 16 node).

Cloudian memberikan ketahanan dan ketersediaan data Anda. HyperStore dapat mencadangkan dan mengarsipkan data Anda, memberi Anda versi yang sangat tersedia untuk mengalirkan pada saat dibutuhkan.

Di HyperStore, penyimpanan terjadi di belakang firewall, Anda dapat menyesuaikan batas geografis untuk mengakses data, dan menentukan kebijakan untuk sinkronisasi data antar perangkat pengguna. HyperStore memberi Anda kemampuan berbagi file berbasis cloud di perangkat lokal, dan kontrol untuk melindungi data Anda di lingkungan cloud apa pun.

7.3 Pengaturan Akses Pengguna

Pengaturan akses pengguna adalah aspek krusial dalam desain dan pengelolaan sistem basis data yang bertujuan untuk melindungi data sensitif dan menjaga integritasnya. Proses ini melibatkan pengaturan hak akses dan kontrol yang tepat untuk mengontrol siapa yang memiliki izin untuk mengakses, memodifikasi, atau menghapus data dalam basis data. Berikut adalah panduan mendalam tentang pengaturan akses pengguna dalam sistem basis data:

- Identifikasi Kebutuhan Akses: Langkah pertama dalam pengaturan akses pengguna adalah memahami kebutuhan akses pengguna dalam organisasi. Identifikasi pengguna dan peran mereka dalam organisasi, serta data apa yang perlu mereka akses dan bagaimana mereka akan menggunakan data tersebut.

- Desain Model Keamanan: Selanjutnya, desain model keamanan yang sesuai dengan kebutuhan bisnis dan persyaratan regulasi. Ini melibatkan pengelompokan pengguna ke dalam peran dan memberikan hak akses yang sesuai dengan peran tersebut. Misalnya, pengguna biasa mungkin memiliki hak akses terbatas, sementara administrator basis data memiliki hak akses penuh.

- Implementasi Otorisasi: Setelah model keamanan direncanakan, langkah selanjutnya adalah mengimplementasikan otorisasi yang sesuai dalam basis data. Ini melibatkan pembuatan dan pengaturan pengguna, peran, dan izin akses menggunakan fitur yang disediakan oleh sistem basis data, seperti perintah GRANT dan REVOKE dalam SQL.

- Manajemen Hak Akses: Pengaturan akses pengguna adalah proses dinamis yang membutuhkan pemantauan dan pembaruan terus-menerus. Manajemen hak akses melibatkan pemeliharaan daftar pengguna, pembaruan hak akses sesuai dengan perubahan dalam organisasi, dan penghapusan hak akses bagi pengguna yang tidak aktif atau tidak lagi memerlukan akses.

- Penyandian dan Enkripsi: Untuk melindungi data sensitif dari akses yang tidak sah, penting untuk menerapkan penyandian dan enkripsi pada tingkat data. Ini memastikan bahwa data tidak dapat dibaca atau dimanipulasi oleh pihak yang tidak memiliki izin akses yang sesuai, bahkan jika mereka berhasil masuk ke dalam basis data.

- Audit dan Pemantauan: Proses pengaturan akses pengguna juga harus mencakup audit dan pemantauan aktivitas pengguna. Ini memungkinkan administrator basis data untuk melacak siapa yang mengakses data, apa yang mereka lakukan dengan data tersebut, dan kapan aktivitas tersebut terjadi. Log audit dapat digunakan untuk mendeteksi ancaman keamanan dan insiden pelanggaran data.

- Kepatuhan Regulasi: Penting untuk memastikan bahwa pengaturan akses pengguna mematuhi regulasi dan kebijakan privasi data yang berlaku, seperti GDPR, HIPAA, atau CCPA. Ini termasuk perlindungan data pribadi, pemantauan akses pengguna, dan pelaporan insiden keamanan sesuai dengan persyaratan hukum.

- Pendidikan dan Pelatihan Pengguna: Akhirnya, pengaturan akses pengguna harus didukung oleh pendidikan dan pelatihan pengguna yang tepat. Pastikan pengguna memahami peran dan tanggung jawab mereka dalam menjaga keamanan dan integritas data, serta prosedur yang harus diikuti saat mengakses atau memanipulasi data dalam basis data.

- Pembatasan Akses Berdasarkan Konteks: Selain memberikan akses ke basis data secara umum, kadang-kadang diperlukan pembatasan akses berdasarkan konteks atau kondisi tertentu. Misalnya, pengguna hanya diizinkan mengakses data tertentu selama jam kerja atau hanya dari lokasi tertentu.

- Manajemen Identitas: Manajemen identitas adalah aspek penting dari pengaturan akses pengguna yang mencakup proses autentikasi pengguna dan pengelolaan identitas mereka. Ini termasuk penggunaan otentikasi multi-faktor, integrasi dengan layanan otentikasi eksternal, dan manajemen siklus hidup pengguna (misalnya, pembuatan, penghapusan, atau non-aktifkan akun pengguna).

- Penyediaan Layanan Data Secara Aman: Dalam lingkungan yang semakin terhubung, pengaturan akses pengguna juga melibatkan penyediaan layanan data secara aman kepada pengguna eksternal atau pihak ketiga. Ini melibatkan penggunaan teknologi seperti API (Application Programming Interface) yang aman dan protokol autentikasi yang kuat untuk memastikan bahwa data hanya diakses oleh pihak yang berwenang.

- Pemulihan Bencana dan Pengaturan Akses: Dalam situasi darurat atau bencana, penting untuk memiliki rencana pemulihan bencana yang mencakup pemulihan akses pengguna. Ini termasuk cadangan dan pemulihan data, serta mekanisme untuk mengembalikan hak akses pengguna yang sesuai setelah pemulihan dilakukan.

- Penanganan Insiden Keamanan: Pengaturan akses pengguna juga melibatkan penanganan insiden keamanan yang melibatkan pelanggaran atau akses yang tidak sah. Ini termasuk investigasi insiden, perbaikan kerentanan, dan perbaikan prosedur keamanan untuk mencegah kejadian serupa di masa depan.

- Penilaian Risiko: Sebagai bagian dari pengaturan akses pengguna, penting untuk melakukan penilaian risiko secara teratur untuk mengidentifikasi potensi kerentanan dan ancaman keamanan. Ini memungkinkan organisasi untuk mengambil langkah-langkah proaktif untuk mengurangi risiko keamanan dan melindungi data mereka.

- Penggunaan Alat dan Teknologi: Terakhir, pengaturan akses pengguna juga melibatkan penggunaan alat dan teknologi yang tepat untuk mendukung manajemen akses pengguna yang efisien. Ini termasuk alat manajemen identitas, sistem manajemen hak akses (IAM), dan alat pemantauan keamanan yang dirancang khusus untuk melindungi sistem basis data.

- Ketersediaan Layanan: Selain memperhatikan keamanan, penting juga untuk mempertimbangkan ketersediaan layanan dalam pengaturan akses pengguna. Ini melibatkan pemantauan ketersediaan sistem dan infrastruktur yang mendukung layanan basis data, serta implementasi strategi toleransi kesalahan dan pemulihan otomatis untuk meminimalkan downtime.
- Skala Akses Pengguna: Dalam lingkungan yang besar atau kompleks, pengelolaan akses pengguna dapat menjadi tantangan yang kompleks. Pertimbangkan strategi pengelompokan pengguna, delegasi hak akses, dan pembagian kerja antara tim administrasi untuk memastikan pengelolaan yang efisien dan efektif.

- Pembatasan Hak Akses Berbasis Fitur: Selain pembatasan akses berdasarkan peran, juga bisa berguna untuk membatasi hak akses berdasarkan fitur atau fungsi tertentu dalam aplikasi. Misalnya, pengguna hanya diizinkan mengakses fungsi tertentu dalam aplikasi atau melihat jenis data tertentu dalam basis data.

- Pengaturan Akses Eksternal: Dalam beberapa kasus, organisasi mungkin perlu memberikan akses kepada pihak eksternal seperti vendor, mitra bisnis, atau klien. Penting untuk memiliki proses dan mekanisme yang tepat untuk mengelola akses eksternal ini, termasuk pembatasan akses yang sesuai dan pemantauan aktivitas pengguna.

- Pelaporan dan Audit: Implementasi pelaporan dan audit yang efektif adalah bagian penting dari pengaturan akses pengguna. Pelaporan aktivitas pengguna dan audit log memungkinkan untuk memeriksa aktivitas pengguna, mendeteksi pelanggaran keamanan, dan memenuhi persyaratan kepatuhan regulasi.

- Kesadaran dan Pelatihan Pengguna: Kesadaran keamanan adalah kunci untuk keberhasilan pengaturan akses pengguna. Pastikan untuk menyediakan pelatihan yang tepat kepada pengguna tentang kebijakan, prosedur, dan praktik terbaik dalam menggunakan data dan mengelola akses mereka.

- Evaluasi Kinerja dan Perbaikan: Terakhir, penting untuk terus melakukan evaluasi kinerja dan perbaikan dalam pengaturan akses pengguna. Evaluasi rutin tentang efektivitas kebijakan dan praktik akses pengguna, serta identifikasi area untuk perbaikan dan peningkatan, akan membantu memperkuat keamanan dan integritas sistem basis data.

- Pemantauan Aktivitas Pengguna: Selain pelaporan dan audit, penting juga untuk aktif memantau aktivitas pengguna dalam sistem basis data. Ini melibatkan penggunaan alat

pemantauan yang canggih untuk mendeteksi pola perilaku yang mencurigakan atau aktivitas yang tidak biasa yang dapat mengindikasikan ancaman keamanan.

- Pembatasan Akses pada Tingkat Data: Selain pembatasan akses pada tingkat aplikasi atau fungsi, ada juga kebutuhan untuk mempertimbangkan pembatasan akses pada tingkat data yang lebih granular. Misalnya, membatasi akses pengguna hanya pada subset tertentu dari data dalam sebuah tabel.

- Mekanisme Pemulihan Akses: Dalam situasi di mana akses pengguna dicabut atau diubah, penting untuk memiliki mekanisme pemulihan yang efektif. Ini melibatkan pencadangan dan menyimpan konfigurasi hak akses pengguna sehingga dapat dipulihkan jika diperlukan.

- Penanganan Konflik Akses: Dalam lingkungan dengan banyak pengguna dan peran yang kompleks, mungkin timbul konflik akses antara pengguna atau peran yang berbeda. Penting untuk memiliki proses dan mekanisme yang ditetapkan untuk menangani konflik ini dengan adil dan efektif.

- Penyesuaian Akses Berdasarkan Perubahan Bisnis: Bisnis seringkali berubah dan berkembang seiring waktu, yang dapat mempengaruhi kebutuhan akses pengguna. Penting untuk memiliki proses yang fleksibel untuk menyesuaikan hak akses pengguna sejalan dengan perubahan bisnis, seperti penambahan atau penghapusan peran pengguna.

- Peninjauan dan Audit Berkala: Selain pemantauan dan pembaruan kontinu, penting juga untuk melakukan peninjauan dan audit berkala terhadap pengaturan akses pengguna. Ini memastikan bahwa kebijakan dan praktik yang ada masih relevan, efektif, dan mematuhi peraturan yang berlaku.

- Penggunaan Teknologi Identifikasi Lanjutan: Di samping otentikasi dan otorisasi tradisional, juga penting untuk mempertimbangkan penggunaan teknologi identifikasi lanjutan seperti biometrik atau token otentikasi untuk mengamankan akses pengguna dengan lebih kuat.

- Kerjasama dengan Tim Keamanan: Kerjasama antara tim pengelolaan basis data dan tim keamanan informasi sangat penting dalam pengaturan akses pengguna yang efektif. Kolaborasi ini memastikan bahwa kebijakan keamanan data dan praktik terbaik dipatuhi dan diterapkan secara konsisten.

Dengan mempertimbangkan dan menerapkan tambahan ini, organisasi dapat memperkuat pengaturan akses pengguna dalam sistem basis data mereka, meningkatkan keamanan, ketersediaan, dan integritas data, serta meminimalkan risiko keamanan yang terkait dengan akses yang tidak sah atau tidak sesuai.

7.4 Audit dan Pemantauan Keamanan

Basis data yang aman dan andal sangat penting di dunia yang berbasis data saat ini. Audit basis data adalah aspek penting dari keamanan data, karena memungkinkan Anda memantau aktivitas pengguna, melacak perubahan, mendeteksi potensi ancaman, dan memastikan kepatuhan terhadap peraturan. Audit juga membantu mengidentifikasi anomali, yang mungkin mengindikasikan aktivitas berbahaya atau tidak sah, dan mengambil tindakan cepat untuk mencegah kerusakan.

Untuk memastikan strategi audit yang kuat dan efektif, Anda harus memahami jenis audit yang tersedia dan menentukan mana yang paling relevan dengan kebutuhan organisasi Anda. Selain itu, Anda perlu menetapkan kebijakan dan prosedur yang jelas, termasuk kontrol akses, penyimpanan, dan pengarsipan, untuk memastikan bahwa jejak audit Anda aman dan andal. Dalam artikel ini, kita akan mempelajari dasar-dasar audit database, jenis audit, mengapa audit itu penting, praktik terbaik untuk penerapan, dan alat yang tersedia untuk mengaudit database Anda.

Apa itu audit basis data?

Audit basis data adalah proses pemantauan dan pencatatan aktivitas pada basis data. Ini melibatkan pelacakan peristiwa basis data, seperti upaya login, modifikasi, dan kueri, dan menyimpan informasi dalam file log atau basis data. Administrator dan konsultan basis data dapat menggunakan data log untuk melacak aktivitas pengguna, memecahkan masalah, mendeteksi pelanggaran keamanan, dan mematuhi peraturan.

Pengembang dan administrator dapat melakukan audit basis data di berbagai tingkatan, termasuk sistem operasi, server basis data, dan aplikasi. Hal ini juga dapat dilakukan untuk database yang berbeda, seperti database relasional, database NoSQL, dan database berorientasi objek.

Audit basis data melibatkan aktivitas pemantauan dalam basis data untuk menjaga integritas data, melindungi dari pelanggaran data, dan mematuhi peraturan. Audit bertujuan untuk mencatat peristiwa untuk akuntabilitas, mengidentifikasi kesalahan atau penipuan, dan pemecahan masalah. Administrator dapat mencapai audit database melalui berbagai metode, termasuk pemicu, log, dan peringatan.

Audit basis data adalah aspek penting dari keamanan basis data, dan penting untuk memantau aktivitas pada basis data secara terus menerus. Administrator basis data harus melakukan peninjauan rutin terhadap log audit untuk mendeteksi anomali atau aktivitas mencurigakan, sehingga memastikan keamanan basis data. Proses audit harus disesuaikan dengan

kebutuhan spesifik organisasi, dan administrator harus memprioritaskan peristiwa yang akan diaudit berdasarkan kepentingannya. Dengan menerapkan praktik terbaik untuk audit basis data, organisasi dapat memastikan integritas dan keamanan data mereka.

Jenis audit basis data

Administrator dapat melakukan beberapa jenis audit basis data untuk memantau berbagai aspek basis data. Ini termasuk:

- Audit pengguna: Audit ini melacak aktivitas pengguna, seperti login, kueri, dan modifikasi. Mereka mengidentifikasi aktivitas tidak sah atau berbahaya, memecahkan masalah, dan menilai kinerja.
- Audit data: Audit ini melacak perubahan pada data, seperti penyisipan, pembaruan, dan penghapusan. Mereka memantau kualitas data, mendeteksi anomali, dan memastikan kepatuhan terhadap peraturan.
- Audit keamanan: Audit ini melacak akses ke objek database, seperti tabel, tampilan, dan prosedur tersimpan. Mereka dapat digunakan untuk memastikan bahwa hanya pengguna yang berwenang yang memiliki akses ke data sensitif dan untuk memantau peran dan izin pengguna.
- Audit kepatuhan: Audit ini melacak aktivitas yang terkait dengan kepatuhan terhadap peraturan, seperti HIPAA, PCI-DSS, dan GDPR. Mereka memantau akses ke data pribadi atau sensitif, mendeteksi pelanggaran keamanan, dan memastikan bahwa data ditangani dengan tepat.

Mengapa audit basis data penting?

Audit basis data penting karena beberapa alasan. Pertama, ini membantu organisasi memastikan integritas dan keamanan data mereka. Dengan memantau aktivitas pengguna dan perubahan data, organisasi dapat mendeteksi pelanggaran keamanan, mengidentifikasi aktivitas tidak sah, dan mencegah kehilangan atau kerusakan data.

Kedua, audit basis data membantu organisasi mematuhi peraturan dan standar industri. Banyak peraturan yang mewajibkan organisasi untuk melacak dan melaporkan akses ke data sensitif, dan kegagalan untuk mematuhinya dapat mengakibatkan denda atau tindakan hukum.

Terakhir, administrator dapat menggunakan audit database untuk meningkatkan kinerja dan memecahkan masalah. Dengan menganalisis data log, organisasi dapat mengidentifikasi hambatan kinerja, melacak kesalahan, dan mengoptimalkan database mereka.

Praktik terbaik untuk menerapkan audit basis data

Untuk menerapkan audit basis data secara efektif, organisasi harus mengikuti beberapa praktik terbaik, seperti:

- Tetapkan tujuan dan sasaran audit yang jelas
- Identifikasi jenis audit yang diperlukan untuk organisasi Anda
- Pilih alat dan teknologi audit yang sesuai
- Tetapkan tanggung jawab untuk tugas audit
- Menetapkan kebijakan dan prosedur audit
- Tinjau log dan laporan audit secara teratur
- Terus memperbarui dan meningkatkan praktik audit

Dengan mengikuti praktik terbaik ini, organisasi dapat memastikan bahwa program audit database mereka efektif dan memberikan nilai bagi organisasi.

Bagaimana melakukan audit basis data

Menerapkan strategi audit basis data memerlukan pendekatan yang bijaksana. Berikut beberapa praktik terbaik yang perlu dipertimbangkan saat menyiapkan sistem audit database:

1. Tentukan persyaratan audit: Mulailah dengan mengidentifikasi jenis audit yang perlu Anda lakukan berdasarkan tujuan organisasi Anda dan persyaratan kepatuhan. Hal ini dapat mencakup pemantauan aktivitas pengguna, pencatatan akses data, dan audit kepatuhan.
2. Pilih alat audit yang tepat: Banyak alat audit yang tersedia untuk database, mulai dari fitur bawaan hingga perangkat lunak pihak ketiga. Pilih alat yang memenuhi persyaratan audit Anda dan dapat memberikan peringatan dan laporan waktu nyata.
3. Tetapkan garis dasar: Tetapkan garis dasar aktivitas rutin untuk database Anda. Ini akan membantu Anda mendeteksi pola atau perilaku tidak biasa yang mungkin mengindikasikan ancaman keamanan.
4. Siapkan peringatan: Konfigurasikan peringatan untuk memberi tahu Anda ketika peristiwa tertentu terjadi, seperti upaya login yang gagal, akses data yang tidak sah, atau modifikasi data.
5. Tinjau log audit: Tinjau log audit secara rutin untuk mendeteksi anomali, aktivitas tidak biasa, atau potensi ancaman keamanan. Ini akan membantu Anda mengidentifikasi pelanggaran keamanan dan mengambil tindakan segera.

Dengan mengikuti praktik terbaik ini, Anda dapat menyiapkan sistem audit database yang efektif yang membantu melindungi data sensitif Anda, memenuhi persyaratan peraturan, dan menjaga integritas database Anda.

Alat untuk audit basis data

Ada beberapa alat yang tersedia untuk mengimplementasikan audit basis data. Ini termasuk:

- Alat audit basis data asli: Banyak sistem manajemen basis data (DBMS) memiliki kemampuan audit bawaan yang dapat memantau dan melacak aktivitas basis data.
- Alat audit pihak ketiga: Ada beberapa alat pihak ketiga yang menyediakan kemampuan audit lebih lanjut, seperti pemantauan, peringatan, dan pelaporan waktu nyata.
- Alat informasi keamanan dan manajemen kejadian (SIEM): Alat SIEM dapat digunakan untuk mengumpulkan, menganalisis, dan melaporkan data dari berbagai sumber, termasuk log basis data. Mereka menyediakan platform terpusat untuk mengelola peristiwa keamanan dan dapat digunakan untuk menghubungkan data di berbagai sistem.

Saat memilih alat untuk audit basis data, penting untuk mempertimbangkan faktor-faktor seperti tingkat detail yang diberikan, kemudahan penggunaan, skalabilitas, dan biaya.

Bab 8 Sistem Basis Data Terdistribusi

8.1 Konsep Sistem Terdistribusi

Sistem Komputasi Terdistribusi adalah sejumlah elemen proses yang terkoneksi melalui jaringan komputer dan saling bekerjasama dalam melakukan suatu tugas. Basis Data Teridistribusi adalah kumpulan basis-basis data yang saling berhubungan secara logika dan tersebar pada sebuah jaringan komputer. Sistem Manajemen Basis Data adalah sebuah sistem software yang mengelola basis data terdistribusi.

Konsep yang terkait dengan database terdistribusi meliputi arsitektur database terdistribusi, tautan database, pemrosesan transaksi, pengembangan aplikasi, dan dukungan rangkaian karakter.

- Arsitektur Basis Data Terdistribusi: Sistem basis data terdistribusi memungkinkan aplikasi mengakses data dari basis data lokal dan jarak jauh. Dalam sistem database terdistribusi homogen , setiap database adalah Oracle Database. Dalam sistem basis data terdistribusi heterogen , setidaknya salah satu basis data bukan merupakan Basis Data Oracle. Basis data terdistribusi menggunakan arsitektur klien/server untuk memproses permintaan informasi.

- Tautan Basis Data: Konsep sentral dalam sistem basis data terdistribusi adalah tautan basis data . Tautan basis data adalah koneksi antara dua server basis data fisik yang memungkinkan klien mengaksesnya sebagai satu basis data logis.

- Administrasi Basis Data Terdistribusi: Administrasi basis data terdistribusi mencakup topik yang berkaitan dengan otonomi situs, keamanan, audit tautan basis data, dan alat administrasi.

- Pemrosesan Transaksi dalam Sistem Terdistribusi: Transaksi adalah unit kerja logis yang dibentuk oleh satu atau lebih pernyataan SQL yang dieksekusi oleh satu pengguna. Sebuah transaksi dimulai dengan pernyataan SQL pertama yang dapat dieksekusi oleh pengguna dan berakhir ketika transaksi tersebut dilakukan atau dibatalkan oleh pengguna tersebut. Transaksi jarak jauh hanya berisi pernyataan yang mengakses satu node jarak jauh. Transaksi terdistribusi berisi pernyataan yang mengakses banyak node.
- Pengembangan Aplikasi Basis Data Terdistribusi: Pengembangan aplikasi pada sistem terdistribusi menimbulkan permasalahan yang tidak dapat diterapkan pada sistem non-terdistribusi.

- Dukungan Rangkaian Karakter untuk Lingkungan Terdistribusi: Database dan klien yang berbeda dapat menggunakan rangkaian karakter yang berbeda dalam lingkungan terdistribusi.

1. Arsitektur Basis Data Terdistribusi

Sistem basis data terdistribusi memungkinkan aplikasi mengakses data dari basis data lokal dan jarak jauh. Dalam sistem database terdistribusi homogen , setiap database adalah Oracle Database. Dalam sistem basis data terdistribusi heterogen , setidaknya salah satu basis data bukan merupakan Basis Data Oracle. Basis data terdistribusi menggunakan arsitektur klien/server untuk memproses permintaan informasi.

- Sistem Basis Data Terdistribusi Homogen: Sistem basis data terdistribusi homogen hanya mencakup basis data Oracle.

- Sistem Basis Data Terdistribusi Heterogen: Sistem basis data terdistribusi heterogen mencakup basis data Oracle dan basis data non-Oracle.

- Arsitektur Basis Data Klien/Server Server: basis data adalah perangkat lunak Oracle yang mengelola basis data, dan klien adalah aplikasi yang meminta informasi dari server. Setiap komputer dalam suatu jaringan adalah sebuah node yang dapat menampung satu atau lebih database. Setiap node dalam sistem database terdistribusi dapat bertindak sebagai klien, server, atau keduanya, bergantung pada situasinya.

a. Sistem Basis Data Terdistribusi Homogen

Sistem database terdistribusi homogen hanya mencakup database Oracle. Tentang sistem basis data terdistribusi homogen sistem basis data terdistribusi homogen adalah jaringan dua atau lebih basis data oracle yang berada pada satu atau lebih sistem. Basis data terdistribusi versus pemrosesan terdistribusi istilah basis data terdistribusi dan pemrosesan terdistribusi berkaitan erat, namun memiliki arti yang berbeda. Basis data terdistribusi versus basis data yang direplikasi istilah sistem basis data terdistribusi dan replikasi basis data saling terkait, namun berbeda.

b. Tentang Sistem Basis Data Terdistribusi Homogen

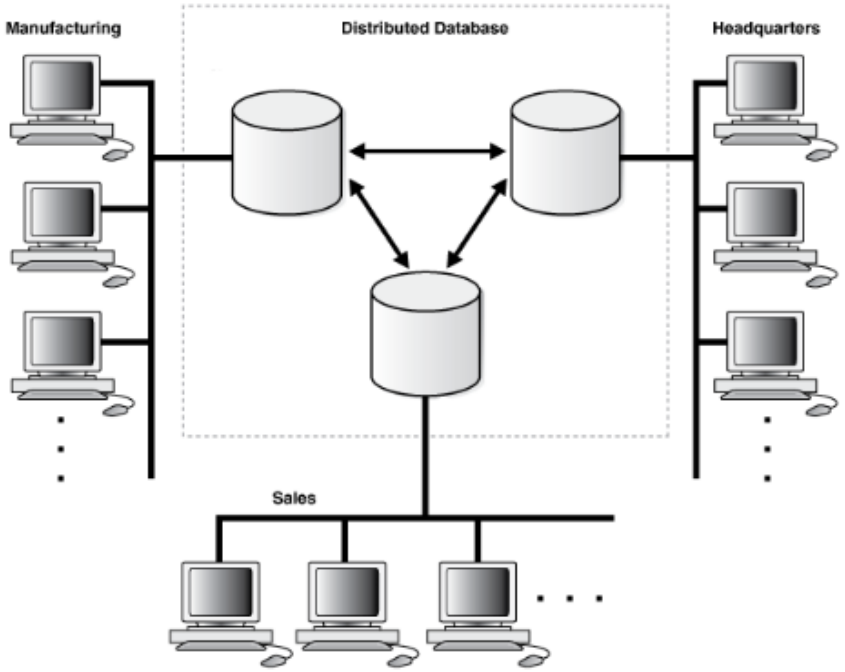
Sistem database terdistribusi homogen adalah jaringan dari dua atau lebih Database Oracle yang berada pada satu atau lebih sistem. Gambar 8.1 mengilustrasikan sistem terdistribusi yang menghubungkan tiga database: hq, mfg, dan sales. Suatu aplikasi dapat secara bersamaan mengakses atau mengubah data di beberapa database dalam satu lingkungan terdistribusi.

Misalnya, satu kueri dari klien Manufaktur di database lokal mfgbisa mengambil data gabungan dari productstabel di database lokal dan depttabel di hqdatabase jarak jauh.

Untuk aplikasi klien, lokasi dan platform database bersifat transparan. Anda juga dapat membuat sinonim untuk objek jarak jauh di sistem terdistribusi sehingga pengguna dapat mengaksesnya dengan sintaksis yang sama dengan objek lokal. Misalnya, jika Anda tersambung ke database mfgtetapi ingin mengakses data di database hq, membuat sinonim mfguntuk depttabel jarak jauh memungkinkan Anda mengeluarkan kueri ini:

PILIH * DARI departemen;

Dengan cara ini, sistem terdistribusi memberikan tampilan akses data asli. Pengguna mfgtidak perlu mengetahui bahwa data yang mereka akses berada di database jarak jauh.



Gambar 8. 1 Contoh Basis Data Terdistribusi Homogen

Sistem database terdistribusi Oracle Database dapat menggabungkan Oracle Database dari rilis yang berbeda. Semua rilis Oracle Database yang didukung dapat berpartisipasi dalam sistem database terdistribusi. Meskipun demikian, aplikasi yang bekerja dengan database terdistribusi harus memahami fungsionalitas yang tersedia di setiap node dalam sistem. Aplikasi database terdistribusi tidak dapat mengharapkan database Oracle7 memahami ekstensi SQL yang hanya tersedia dengan Oracle Database.

c. Basis Data Terdistribusi Versus Pemrosesan Terdistribusi

Istilah database terdistribusi dan pemrosesan terdistribusi sangat erat kaitannya, namun memiliki arti yang berbeda. Ada definisinya sebagai berikut:

- Basis data terdistribusi

Sekumpulan database dalam sistem terdistribusi yang dapat muncul pada aplikasi sebagai sumber data tunggal.

- Pemrosesan terdistribusi

Operasi yang terjadi saat aplikasi mendistribusikan tugasnya di antara komputer berbeda dalam jaringan. Misalnya, aplikasi database biasanya mendistribusikan tugas presentasi front-end ke komputer klien dan memungkinkan server database back-end mengelola akses bersama ke database. Akibatnya, sistem pemrosesan aplikasi basis data terdistribusi lebih sering disebut sebagai sistem aplikasi basis data klien/server.

Sistem database terdistribusi menggunakan arsitektur pemrosesan terdistribusi. Misalnya, server Oracle Database bertindak sebagai klien ketika meminta data yang dikelola oleh server Oracle Database lain.

d. Basis Data Terdistribusi versus Basis Data yang Direplikasi

Istilah sistem basis data terdistribusi dan replikasi basis data saling terkait, namun berbeda. Dalam database terdistribusi murni (yaitu, tidak direplikasi), sistem mengelola satu salinan semua data dan objek database pendukung. Biasanya, aplikasi database terdistribusi menggunakan transaksi terdistribusi untuk mengakses data lokal dan jarak jauh serta memodifikasi database global secara real-time.

Istilah replikasi mengacu pada operasi penyalinan dan pemeliharaan objek database di beberapa database milik sistem terdistribusi. Meskipun replikasi bergantung pada teknologi basis data terdistribusi, replikasi basis data menawarkan manfaat aplikasi yang tidak mungkin dilakukan dalam lingkungan basis data terdistribusi murni.

Paling umum, replikasi digunakan untuk meningkatkan kinerja database lokal dan melindungi ketersediaan aplikasi karena terdapat opsi akses data alternatif. Misalnya, suatu aplikasi biasanya mengakses database lokal daripada server jarak jauh untuk meminimalkan lalu lintas jaringan dan mencapai kinerja maksimum. Selain itu, aplikasi dapat terus berfungsi jika server lokal mengalami kegagalan, namun server lain dengan data yang direplikasi tetap dapat diakses.

2. Sistem Basis Data Terdistribusi Heterogen

Sistem database terdistribusi heterogen mencakup database Oracle dan database non-Oracle.

- Tentang Sistem Basis Data Terdistribusi Heterogen
  Dalam sistem basis data terdistribusi heterogen, setidaknya salah satu basis datanya adalah sistem Basis Data non-Oracle. Untuk aplikasinya, sistem database terdistribusi heterogen muncul sebagai Oracle Database lokal tunggal. Server Oracle Database lokal menyembunyikan distribusi dan heterogenitas data.
- Layanan Heterogen
  Layanan Heterogen (HS) adalah komponen terintegrasi dalam server Oracle Database dan teknologi yang memungkinkan rangkaian produk Oracle Transparent Gateway saat ini.
- Agen Gateway Transparan
  Untuk setiap sistem Database non-Oracle yang Anda akses, Layanan Heterogen dapat menggunakan agen gateway transparan untuk berinteraksi dengan sistem Database non-Oracle yang ditentukan. Agen ini khusus untuk sistem Database non-Oracle, sehingga setiap jenis sistem memerlukan agen yang berbeda.
- Konektivitas Generik
  Konektivitas generik memungkinkan Anda terhubung ke penyimpanan data Database non-Oracle dengan menggunakan agen ODBC Layanan Heterogen atau agen DB OLE Layanan Heterogen.

a. Tentang Sistem Basis Data Terdistribusi Heterogen

Dalam sistem database terdistribusi heterogen, setidaknya salah satu database adalah sistem Database non-Oracle. Untuk aplikasinya, sistem database terdistribusi heterogen muncul sebagai Oracle Database lokal tunggal. Server Oracle Database lokal menyembunyikan distribusi dan heterogenitas data.

Server Oracle Database mengakses sistem Database non-Oracle menggunakan Oracle Heterogeneous Services dengan agen . Jika Anda mengakses penyimpanan data non-Oracle Database menggunakan Oracle Transparent Gateway, maka agen tersebut adalah aplikasi khusus sistem. Misalnya, jika Anda menyertakan database Sybase dalam sistem terdistribusi Oracle Database, maka Anda harus mendapatkan gateway transparan khusus Sybase agar Oracle Database di sistem dapat berkomunikasi dengannya.

b. Pelayanan Heterogen

Heterogeneous Services (HS) adalah komponen terintegrasi dalam server Oracle Database dan teknologi yang memungkinkan rangkaian produk Oracle Transparent Gateway saat ini.

HS menyediakan arsitektur umum dan mekanisme administrasi untuk produk gateway Oracle Database dan fasilitas akses heterogen lainnya. Selain itu, ia menyediakan fungsionalitas yang kompatibel bagi pengguna sebagian besar rilis Oracle Transparent Gateway sebelumnya.

c. Agen Gerbang Transparan

Untuk setiap sistem Database non-Oracle yang Anda akses, Layanan Heterogen dapat menggunakan agen gateway transparan untuk berinteraksi dengan sistem Database non-Oracle yang ditentukan. Agen ini khusus untuk sistem Database non-Oracle, sehingga setiap jenis sistem memerlukan agen yang berbeda.

Agen gateway transparan memfasilitasi komunikasi antara Oracle Database dan sistem non-Oracle Database dan menggunakan komponen Layanan Heterogen di server Oracle Database. Agen mengeksekusi permintaan SQL dan transaksional di sistem Database non-Oracle atas nama server Oracle Database.

d. Konektivitas Generik

Konektivitas generik memungkinkan Anda terhubung ke penyimpanan data non-Oracle Database dengan menggunakan agen ODBC Layanan Heterogen atau agen DB OLE Layanan Heterogen.

Keduanya disertakan dengan produk Oracle Anda sebagai fitur standar. Sumber data apa pun yang kompatibel dengan standar ODBC atau OLE DB dapat diakses menggunakan agen konektivitas generik.

Keuntungan konektivitas generik adalah Anda mungkin tidak perlu membeli dan mengonfigurasi agen khusus sistem yang terpisah. Anda menggunakan driver ODBC atau OLE DB yang dapat berinteraksi dengan agen. Namun, beberapa fitur akses data hanya tersedia dengan agen gateway transparan.

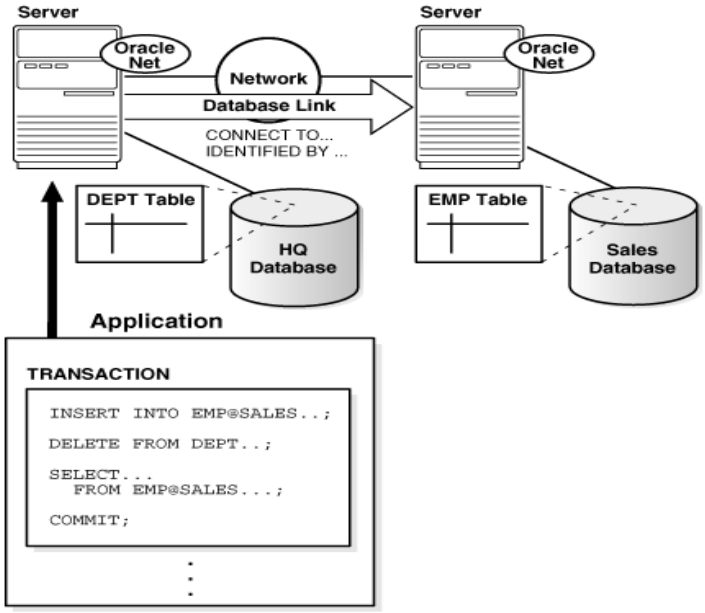
3. Arsitektur Basis Data Klien/Server

Server basis data adalah perangkat lunak Oracle yang mengelola basis data, dan klien adalah aplikasi yang meminta informasi dari server.

Setiap komputer dalam suatu jaringan adalah sebuah node yang dapat menampung satu atau lebih database.

Setiap node dalam sistem database terdistribusi dapat bertindak sebagai klien, server, atau keduanya, bergantung pada situasinya.

Pada Gambar 8.2 , host database hqbertindak sebagai server database ketika sebuah pernyataan dikeluarkan terhadap data lokalnya (misalnya, pernyataan kedua dalam setiap transaksi mengeluarkan pernyataan terhadap depttabel lokal), namun bertindak sebagai klien ketika mengeluarkan pernyataan terhadap data jarak jauh (misalnya, pernyataan pertama dalam setiap transaksi dikeluarkan terhadap tabel jarak jauh empdi salesdatabase).



Gambar 8. 2 Contoh Sistem Database Terdistribusi Database Oracle

Klien dapat terhubung secara langsung atau tidak langsung ke server database. Koneksi langsung terjadi ketika klien terhubung ke server dan mengakses informasi dari database yang terdapat di server tersebut. Misalnya, jika Anda terhubung ke hqdatabase dan mengakses depttabel pada database ini seperti pada Gambar 8.2 , Anda dapat mengeluarkan perintah berikut:

PILIH * DARI departemen;

Kueri ini bersifat langsung karena Anda tidak mengakses objek pada database jarak jauh.

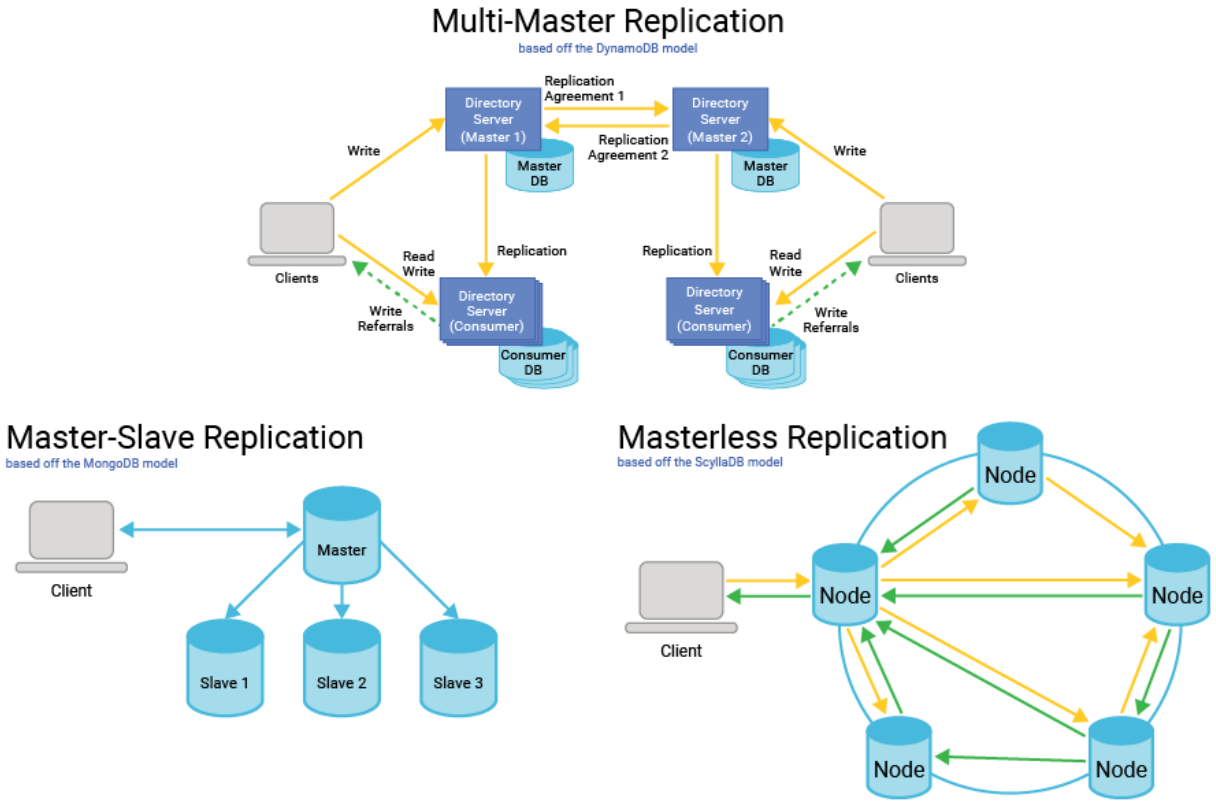
Sebaliknya, koneksi tidak langsung terjadi ketika klien terhubung ke server dan kemudian mengakses informasi yang terdapat dalam database di server berbeda. Misalnya, jika Anda terhubung ke hqdatabase tetapi mengakses emptabel di salesdatabase jarak jauh seperti pada Gambar 8. 2 , Anda dapat mengeluarkan perintah berikut:

PILIH * DARI emp@sales;

Kueri ini tidak langsung karena objek yang Anda akses tidak ada dalam database yang terhubung langsung dengan Anda.

8.2 Replikasi Data

Replikasi basis data adalah penyalinan data yang berkelanjutan di seluruh node dalam cluster basis data terdistribusi . Tujuannya adalah untuk meningkatkan keandalan sistem, toleransi kesalahan, dan aksesibilitas data.



Gambar 8. 3 Contoh Replikasi Data

Apa itu Replikasi Basis Data

Replikasi database terdistribusi melibatkan penyalinan data di beberapa node sehingga satu node yang down tidak membuat sistem tidak tersedia dan/atau mengakibatkan hilangnya data.

Basis data NoSQL cenderung mendukung replikasi data menggunakan tiga model dasar: basis data multi-master, seperti DynamoDB, arsitektur master-slave, seperti MongoDB, dan tanpa master, seperti ScyllaDB. Mengingat ketergantungan mereka pada node master, arsitektur multimaster dan master-slave menimbulkan titik kegagalan. Ketika seorang master turun, proses pemilihan master baru menyebabkan waktu henti yang singkat. Meskipun penundaannya

mungkin minimal, diukur dalam milidetik, penundaan tersebut masih dapat menyebabkan pelanggaran SLA.

Arsitektur tanpa master mengatasi keterbatasan ini. Dalam database ini, data direplikasi ke beberapa node, yang semuanya sama. Dalam arsitektur tanpa master, tidak ada satu node pun yang dapat melumpuhkan seluruh cluster. Topologi tanpa master pada umumnya melibatkan tiga atau lebih replika untuk setiap kumpulan data. Mengadopsi database NoSQL yang mengimplementasikan arsitektur tanpa master memberikan lapisan ketahanan lain untuk aplikasi bervolume tinggi dan berlatensi rendah.

Metode Replikasi Basis Data

Ada dua cara untuk mengatur waktu transfer data dalam proses replikasi data: replikasi asinkron dan sinkron.

Replikasi asinkron dalam database NoSQL membuat klien mengirim data ke server model tempat replika mengambil data. Server model mengirim ping ke klien untuk mengonfirmasi penerimaan data dan menyalinnya ke replika. Replikasi database asinkron menawarkan kemudahan penggunaan dan fleksibilitas, namun menambah risiko kehilangan data yang tidak diketahui.

Replikasi sinkron menyalin data ke server model dari server klien dan kemudian mereplikasinya ke semua server replika sebelum memberi tahu klien. Hal ini lebih memakan waktu untuk verifikasi dan lebih kaku, namun lebih mungkin untuk berhasil menyalin semua data sebelum melanjutkan.

Jenis replikasi database juga dapat diklasifikasikan berdasarkan arsitektur server. Arsitektur pemimpin tunggal secara longgar dianalogikan dengan versi replikasi data sinkron yang agak tidak fleksibel. Arsitektur multi-pemimpin dan arsitektur tanpa pemimpin lebih mirip dengan versi terdistribusi dengan banyak server dan replikasi database asinkron.

Replikasi database SQL dapat digambarkan sebagai replikasi transaksional, snapshot, dan penggabungan.

Replikasi transaksional. Dalam hal ini, DDBMS mereplikasi transaksi atau perubahan pada database asli secara berurutan secara real-time dan pengguna merasakan perubahan ini pada database yang direplikasi hampir seketika. Replikasi transaksional menjamin konsistensi transaksional dan biasanya digunakan dalam lingkungan server-ke-server.

Replikasi cuplikan. DDBMS menangkap dan menimpa snapshot data dari database asli ke database penerima menggunakan server yang sama. Replikasi snapshot tidak memantau pembaruan data dan mendistribusikan data persis seperti yang terlihat pada saat itu. Replikasi

snapshot sedikit lebih lambat dibandingkan replikasi transaksional dan biasanya digunakan ketika perubahan data jarang terjadi.

Gabungkan replikasi. Jenis replikasi yang paling kompleks, kasus penggunaan ini melihat DDBMS menggabungkan data dari beberapa database ke dalam satu database penerima. Replikasi penggabungan memungkinkan perubahan independen oleh penerbit dan pelanggan dan memungkinkan satu penerbit mengirimkan perubahan ke beberapa pelanggan. Replikasi penggabungan sering digunakan di lingkungan server-ke-klien. Replikasi penuh, replikasi parsial, dan tanpa skema replikasi semuanya dimungkinkan.

Replikasi penuh, di mana setiap situs menyimpan seluruh database, menawarkan beberapa keuntungan, termasuk ketersediaan data yang tinggi, peningkatan kinerja dari pengambilan data lokal, dan eksekusi kueri yang lebih cepat. Kerugian dari replikasi penuh mencakup proses pembaruan yang lebih lambat yang diperlukan di lokasi berbeda dan konkurensi lebih sulit dicapai.

Keuntungan dari replikasi parsial, yaitu hanya fragmen database yang sering digunakan yang direplikasi, adalah bahwa replikasi ini menentukan berapa banyak salinan dari setiap bagian data yang akan dibuat berdasarkan seberapa penting data tersebut.

Keuntungan tanpa replikasi, yang menyimpan setiap fragmen hanya di satu situs, adalah kemudahan mencapai konkurensi dan pemulihan data. Kerugiannya termasuk eksekusi kueri yang lebih lambat dan data yang kurang mudah tersedia.

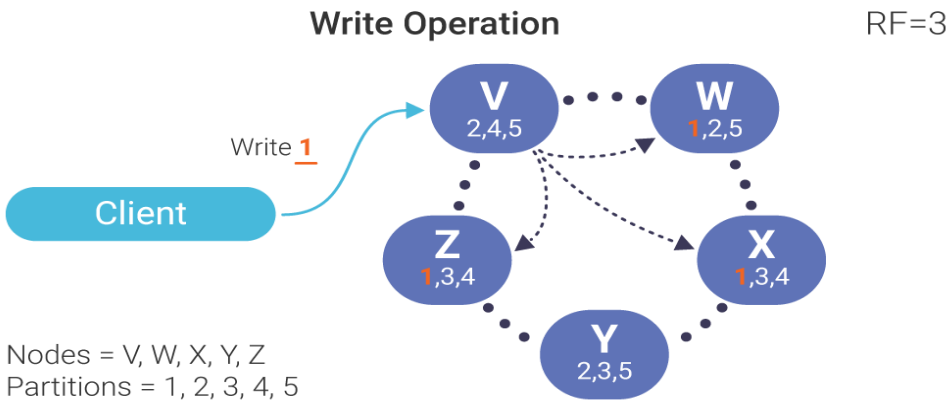
Bagaimana Cara Kerja Replikasi Database NoSQL

Basis data non-relasional mendukung replikasi data secara asli dengan mengikuti beberapa model dasar. Namun, sebagian besar dari hal ini menimbulkan titik kegagalan pada satu database utama. Misalnya, database multi-primer, seperti DynamoDB, dan arsitektur primer-sekunder, seperti MongoDB, keduanya menimbulkan kelemahan ini. Ketika database utama gagal, downtime dan pelanggaran SLA mungkin terjadi.

Struktur tanpa database utama seperti ScyllaBD, yang mereplikasi data di banyak node, mengatasi keterbatasan ini. Jenis arsitektur ini, juga terlihat pada replikasi database Cassandra, menyediakan banyak replika untuk setiap kumpulan data dan memastikan tidak ada satu node pun yang dapat menyebabkan kegagalan seluruh cluster.

Contoh Replikasi Database NoSQL

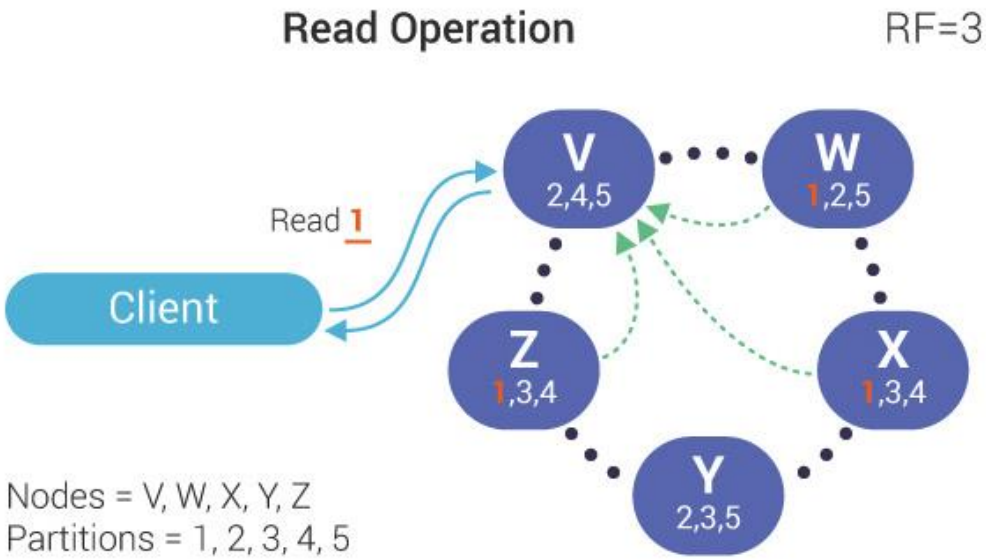
Dalam database seperti Cassandra atau ScyllaDB,dData selalu direplikasi secara otomatis. Operasi baca atau tulis dapat terjadi pada data yang disimpan di node mana pun yang direplikasi.



Gambar 8. 4 Contoh Replikasi Database NoSQL Write Operation

Pada contoh di atas, klien kami mengirimkan permintaan untuk menulis partisi 1 ke node V; Data 1 direplikasi ke node W, X, dan Z.

Kita mempunyai Faktor Replikasi (RF) sebesar 3. Dalam gambar ini, V adalah node koordinator tetapi bukan node replikator. Namun, node replikator juga dapat menjadi node koordinator, dan sering kali memang demikian.



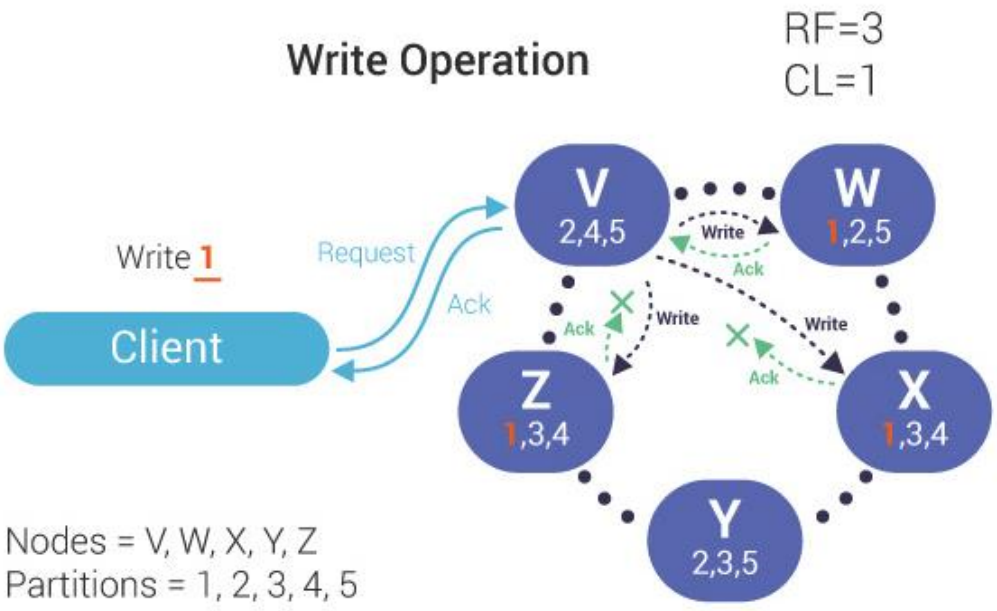
Gambar 8. 5 Contoh Replikasi Database NoSQL Read Operation

Selama operasi baca, klien mengirimkan permintaan ke koordinator. Efektif karena RF=3, 3 node merespons permintaan baca.

Tingkat Konsistensi (CL) menentukan berapa banyak replika dalam sebuah klaster yang harus mengakui operasi baca atau tulis sebelum dianggap berhasil.

Terlepas dari Tingkat Konsistensi, penulisan selalu dikirim ke semua replika, sebagaimana diatur oleh Faktor Replikasi. Tingkat Konsistensi mengontrol kapan klien mengakui, bukan berapa banyak replika yang diperbarui.

Selama operasi tulis, koordinator berkomunikasi dengan replika (jumlahnya bergantung pada Faktor Replikasi). Penulisan berhasil ketika jumlah replika yang ditentukan mengonfirmasi penulisan.



Gambar 8. 6 Contoh Hasil Replikasi Database NoSQL Write Operation

Dalam diagram di atas, panah ganda menunjukkan permintaan operasi tulis yang masuk ke koordinator dari klien dan pengakuan dikembalikan. Karena Tingkat Konsistensinya satu, koordinatornya, V, hanya harus menunggu tulisan dikirim dan direspon oleh satu node di cluster yaitu W.

Karena RF=3, partisi 1 kita juga ditulis ke node X dan Z, namun koordinator tidak perlu menunggu tanggapan dari mereka untuk mengonfirmasi operasi penulisan yang berhasil. Dalam praktiknya, ucapan terima kasih dari node X dan Z dapat sampai ke koordinator di lain waktu, setelah koordinator memberi ucapan terima kasih kepada klien.

Alat Replikasi Basis Data

Organisasi biasanya menggunakan alat replikasi database atau kemampuan bawaan yang dikembangkan oleh pembuat perangkat lunak database atau berinvestasi dalam solusi atau perangkat lunak replikasi database pihak ketiga untuk mengelola dan melaksanakan proses replikasi database.

Berbagai platform perangkat lunak replikasi database pihak ketiga dan alat lainnya bekerja dengan berbagai database populer:

- Replikasi Data Informatica menargetkan berbagai peralatan gudang data dan database untuk menawarkan rangkaian produk untuk integrasi, streaming, kualitas, dan penyembunyian data perusahaan.
- Quest SharePlex sebagian besar berfokus pada solusi lokal dan cloud untuk replikasi database Oracle .
- Perangkat lunak Qlik Replikasi menggunakan penangkapan aliran data berkelanjutan dan otomatisasi berbasis log untuk meminimalkan beban kerja operasi TI dan memproses data besar secara efisien.

Contoh alat untuk replikasi database dari vendor antara lain sebagai berikut:

- Alat replikasi IBM Db2 SQL memiliki opsi replikasi Q dan SQL untuk mendistribusikan data sumber ke beberapa target, namun tidak ideal untuk semua penggunaan karena latensi tinggi.
- Termasuk Layanan Integrasi SQL Server (SSIS) untuk replikasi database Azure , integrasi SQL Microsoft dilengkapi alat untuk membersihkan, menggabungkan, menggabungkan, menyalin, mengekstraksi, dan memuat data.
- Alat Oracle GoldenGate menawarkan konfigurasi sederhana, manajemen dan pemantauan visual, serta pengambilan berbasis log untuk database Oracle.

Pencadangan Basis Data vs Replikasi

Pencadangan basis data—terkadang disebut pencerminan—dan replikasi basis data tidaklah sama. Pencadangan basis data melibatkan penyimpanan salinan database utama yang direplikasi sebagai tindakan pencegahan keamanan jika terjadi kegagalan. Sebaliknya, replikasi database bukan sekedar rencana cadangan namun melibatkan ketersediaan data yang lebih tinggi dan efisiensi operasional yang lebih besar melalui proses yang berkelanjutan.

Sharding Basis Data vs Replikasi

Sharding basis data tidak mereplikasi data. Sebaliknya, ia memecah satu database menjadi beberapa bagian dan menyimpannya di sistem yang berbeda. Berbeda dengan solusi replikasi database, sharding saja tidak dapat menghasilkan ketersediaan yang tinggi.

Keuntungan Replikasi Database

Ada beberapa kelebihan dan kekurangan replikasi data dalam database terdistribusi, termasuk konsistensi trade-off antara konsistensi akhir dan konsistensi yang kuat. Replikasi data memindahkan data dengan aman dan efisien untuk meningkatkan ketersediaan dan kinerja aplikasi dan database.

Keuntungan utama menggunakan replikasi dalam sistem database terdistribusi adalah jika sebuah node down, faktor replikasi yang lebih tinggi berarti probabilitas yang lebih tinggi bahwa data pada node tersebut ada di salah satu node yang tersisa. Misalnya, jika Anda memiliki RF 3 dan 2 node turun, satu node akan tetap tersedia.

Apakah ScyllaDB Menawarkan Opsi untuk Replikasi Basis Data

ScyllaDB menghadirkan replikasi data database multi-pusat data otomatis, yang menyinkronkan cluster di seluruh zona ketersediaan geografis. Pengguna menetapkan faktor replikasi (RF) dan ScyllaDB menyimpan banyak salinan data yang sama di seluruh cluster pada beberapa node. Dengan cara ini, meskipun sebuah node hilang, pengguna masih dapat menemukan data yang berada di suatu tempat di cluster.

Dalam klaster multi-pusat data, data disalin secara asinkron antar klaster sesuai dengan cara ruang kunci didefinisikan. Misalnya dengan replikasi DC1:3, data hanya akan disimpan di pusat data pertama pada tiga node berbeda. Mengubah faktor replikasi menjadi “DC1:3, DC2:3”, ScyllaDB akan menyimpan tiga salinan data di setiap pusat data. Di ScyllaDB, replikasi multi-pusat data bersifat rack-aware dan datacenter-aware, memastikan bahwa replika didistribusikan baik secara fisik maupun geografis.

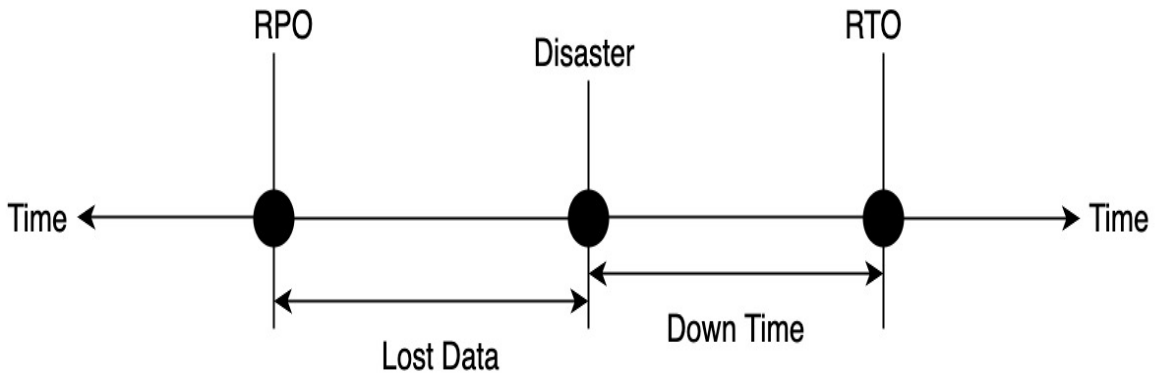
8.3 Pemulihan Bencana untuk Basis Data

Pemulihan bencana (DR) adalah fitur inti database tingkat perusahaan. Vendor database terus berupaya meningkatkan DR, dan dalam 10 tahun terakhir mereka telah membuat inovasi besar.

Mengukur HA dan DR

Tujuan HA dan DR adalah menjaga sistem tetap berjalan pada tingkat operasional. Keduanya mencoba menghilangkan satu titik kegagalan dalam sistem dan mengotomatiskan proses failover atau pemulihan.

Ketersediaan yang tinggi biasanya diukur dengan persentase waktu pengoperasian sistem per tahun. Pemulihan bencana berfokus pada mengembalikan sistem ke layanan setelah bencana dengan kehilangan data minimal. Hal ini diukur dengan dua metrik: waktu yang diperlukan untuk pemulihan atau Sasaran Waktu Pemulihan (RTO), dan hilangnya volume data, atau Sasaran Titik Pemulihan (RPO). RPO dan RTO harus dikurangi semaksimal mungkin.



Gambar 8. 7 Contoh HA/DR dan pengukurannya

Setiap bencana bersifat unik, sehingga Fault Tolerance Target (FTT) menggambarkan cakupan maksimum bencana yang dapat dipertahankan oleh sistem. FTT yang umum digunakan adalah tingkat wilayah, yang menunjukkan bahwa suatu bencana berdampak pada wilayah geografis seperti negara bagian atau kota.

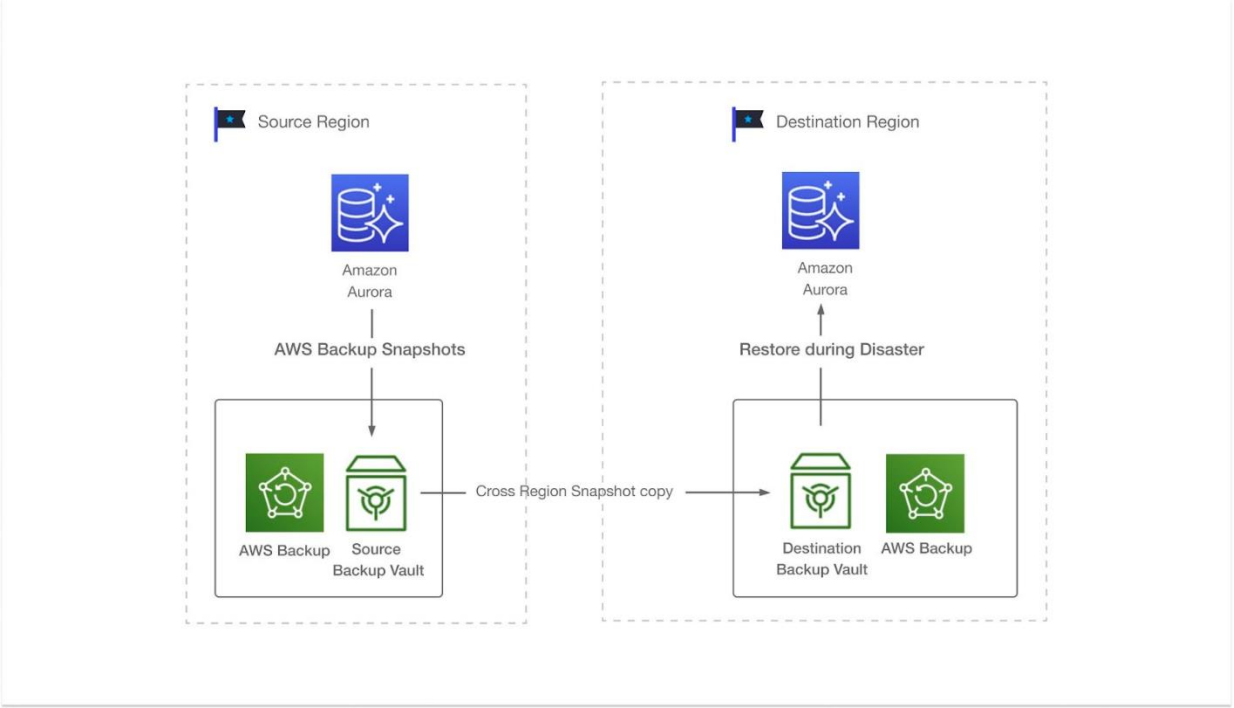
Sejarah singkat DR

Teknologi database DR telah melalui tiga tahap: pencadangan dan pemulihan, aktif-pasif, dan multi-aktif.

1. Cadangkan dan pulihkan

Pada tahap awal pemulihan bencana, database memanfaatkan blok data dan log transaksi untuk membuat cadangan data lengkap dan bertahap. Jika terjadi bencana, database dipulihkan dari cadangan dan log transaksi aplikasi.

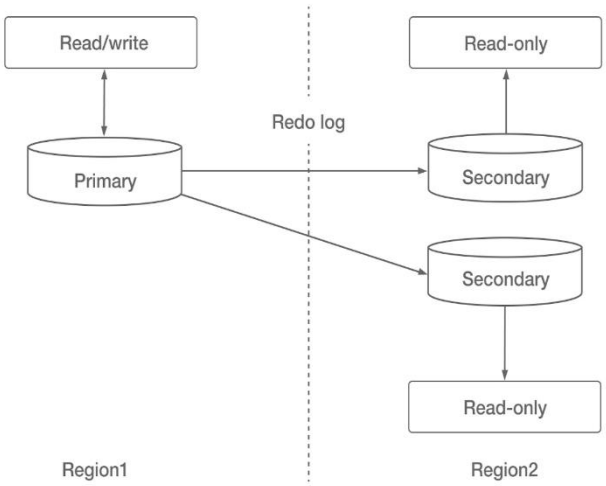
Dalam beberapa tahun terakhir, layanan database cloud publik telah menggabungkan replikasi penyimpanan dengan teknologi pencadangan database tradisional dan menawarkan pencadangan pemulihan otomatis lintas wilayah berdasarkan snapshot. Pendekatan ini secara teratur menghasilkan snapshot dari database di wilayah sumber dan mereplikasi file snapshot ke wilayah tujuan. Jika wilayah sumber mengalami error, database akan dipulihkan dari wilayah tujuan, dan layanan akan dilanjutkan. RTO dan RPO solusi ini dapat berdurasi hingga beberapa jam, sehingga paling sesuai untuk aplikasi yang tidak memiliki persyaratan ketersediaan yang ketat.



Gambar 8. 8 Contoh Pendekatan pencadangan dan pemulihan DR

2. DR aktif-pasif

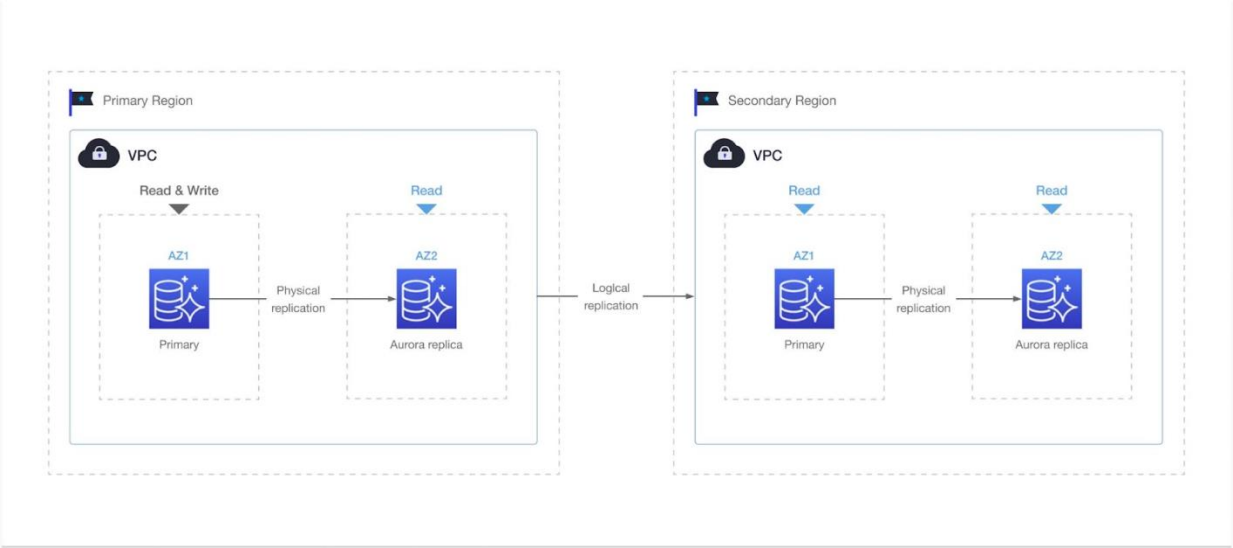
Cluster database menandai tahap kedua pengembangan. Dalam sebuah cluster, node master membaca dan menulis data. Satu atau lebih node cadangan menerima log transaksi dan menerapkannya, sehingga memberikan kemampuan membaca dengan beberapa penundaan.



Gambar 8. 9 Contoh DR aktif-pasif

Meskipun solusi ini melibatkan konsep cluster, namun tetap didasarkan pada database monolitik. Skalabilitas terbatas pada operasi baca; itu tidak dapat menskalakan penulisan. Tentu saja, dibandingkan pendahulunya, RTO dikurangi menjadi beberapa menit dan RPO beberapa detik.

Amazon Aurora, yang menggunakan replikasi logis dengan replika baca lintas wilayah, adalah salah satu layanan database cloud awal yang dibangun berdasarkan teknologi ini.



Gambar 8. 10 Contoh Replikasi logis Aurora dengan replika baca lintas wilayah

Dalam beberapa tahun terakhir, Aurora telah mengembangkan desain ini dan menawarkan layanan Database Global. Layanan ini menggunakan teknologi replikasi penyimpanan untuk mereplikasi data secara asinkron dari wilayah sumber ke wilayah tujuan. Jika wilayah sumber tidak berfungsi, layanan dapat segera melakukan failover ke kluster cadangan. RTO dapat dikurangi menjadi beberapa menit, dan RPO kurang dari 1 detik.

3. Pemulihan bencana multi-aktif

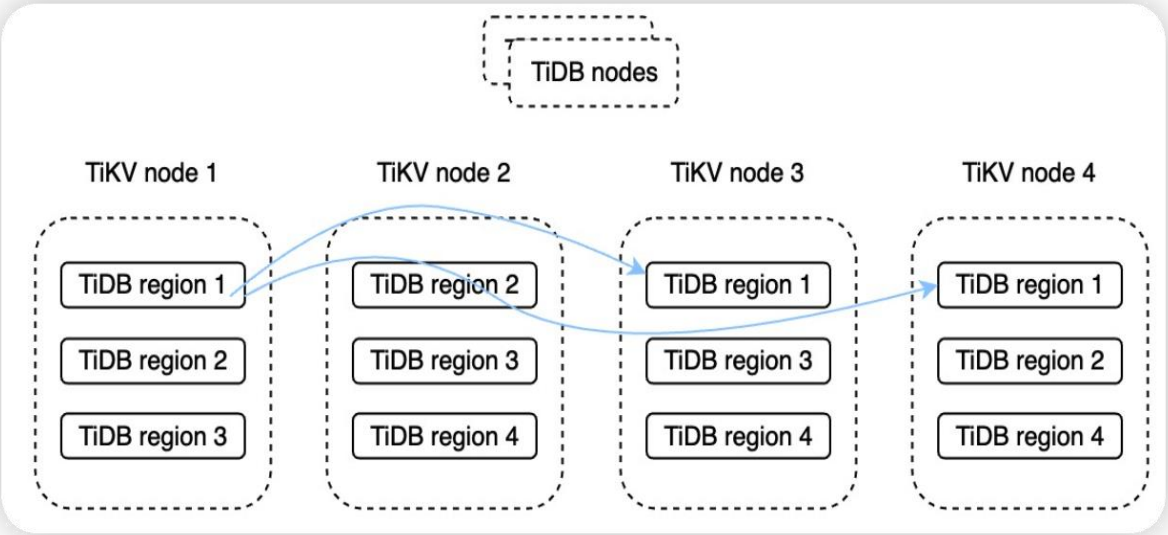
Dalam pemulihan bencana multi-aktif, database menyediakan setidaknya tiga node layanan baca dan tulis untuk salinan data yang sama, dan database dapat melakukan penskalaan atau penskalaan berdasarkan beban kerja. Persyaratan di balik kemampuan ini berasal dari meluasnya aplikasi skala internet, yang memerlukan kinerja lebih tinggi, latensi lebih rendah, ketersediaan lebih tinggi, skalabilitas elastis, dan elastisitas dari database.

Multi-aktivitas terbentuk dengan database sharding tradisional, yang berbagi data berdasarkan satu atau beberapa kolom. Solusi sharding mencapai DR melalui replikasi log transaksi. Misalnya, Google pernah memelihara sistem sharding MySQL yang sangat besar. Solusi ini menawarkan beberapa tingkat skalabilitas, namun tidak dapat meningkatkan kemampuan pemulihan bencana seiring bertambahnya pecahan. Kinerja akan menurun secara signifikan dan biaya pemeliharaan akan melonjak. Oleh karena itu, sharding adalah solusi transisi untuk multi-aktivitas.

Dalam beberapa tahun terakhir, database shared-nothing berdasarkan protokol konsensus Raft atau Paxos berkembang pesat. Mereka memecahkan tantangan skalabilitas dan ketersediaan yang disebutkan di atas. Pemain kunci di era multi-aktif termasuk TiDB dan CockRoachDB. Basis data mereka, bersama dengan teknologi DR mereka, membuat sebagian besar basis data lama dan layanan basis data relasional (RDS) menjadi usang.

DR multi-aktif dengan database terdistribusi

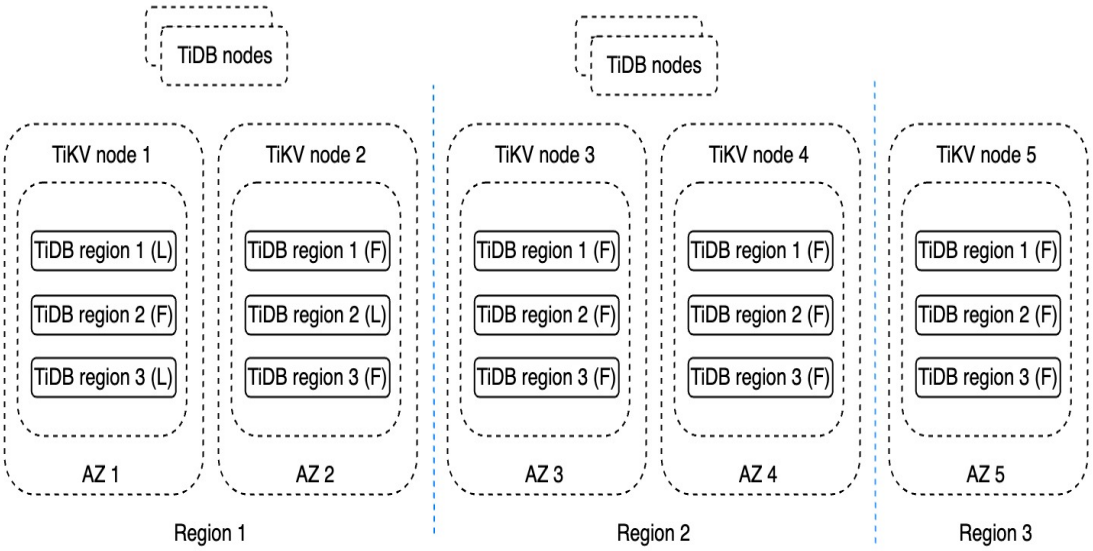
Mari kita lihat DR multi-aktif yang diterapkan pada database terdistribusi. Misalnya, TiDB adalah database terdistribusi open source dan sangat tersedia. Ini membagi setiap tabel atau partisi menjadi wilayah TiDB yang lebih kecil dan menyimpan banyak salinan data di wilayah TiDB tersebut pada node TiKV yang berbeda. Ini disebut redundansi data. TiDB mengadopsi protokol konsensus Raft, jadi ketika data berubah, komitmen transaksi hanya dikembalikan ketika log transaksi disinkronkan ke sebagian besar salinan data. Ini sangat meningkatkan RPO basis data. Faktanya, dalam banyak kasus, RPO adalah 0. Hal ini memastikan konsistensi data. Selain itu, arsitektur TiDB memisahkan mesin penyimpanan dan komputasinya. Hal ini memungkinkan pengguna untuk memasukkan atau mengeluarkan node TiDB dan node penyimpanan TiKV seiring perubahan beban kerja mereka.



Gambar 8. 11 Contoh Arsitektur penyimpanan TiDB

Solusi DR multi-Wilayah yang khas

Diagram berikut menunjukkan bagaimana TiDB memberikan solusi pemulihan bencana multi-Wilayah pada umumnya.



Gambar 8. 12 Contoh Solusi pemulihan bencana multi-wilayah TiDB

Ini adalah istilah kunci untuk arsitektur TiDB DR:

- Wilayah TiDB : penjadwalan dan unit penyimpanan TiDB.
- Wilayah : dua situs atau kota.
- Availability zone (AZ) : zona HA independen. Dalam kebanyakan kasus, AZ adalah dua pusat data atau kota yang letaknya lebih dekat satu sama lain dalam suatu Wilayah.
- L : replika Leader dalam grup Rakit
- F : replika Follower di grup Raft

Pada gambar di atas, setiap Wilayah berisi dua replika data. Mereka berlokasi di AZ yang berbeda, dan keseluruhan cluster mencakup tiga Wilayah. Wilayah 1 biasanya memproses permintaan baca dan tulis. Wilayah 2 digunakan untuk failover setelah terjadi bencana di Wilayah 1, dan juga dapat memproses beberapa beban baca yang tidak sensitif terhadap latensi. Wilayah 3 merupakan replika yang menjamin konsensus tetap dapat tercapai meskipun Wilayah 1 sedang down. Konfigurasi tipikal ini disebut sebagai arsitektur “2-2-1”. Arsitektur ini tidak hanya menjamin pemulihan bencana, namun memberikan kemampuan multi-aktif pada bisnis. Dalam arsitektur ini:

- Target toleransi kesalahan terbesar bisa berada di tingkat Wilayah.
- Kemampuan menulis dapat diperluas.
- RPO-nya adalah 0.
- RTO dapat diatur ke satu menit atau bahkan kurang.

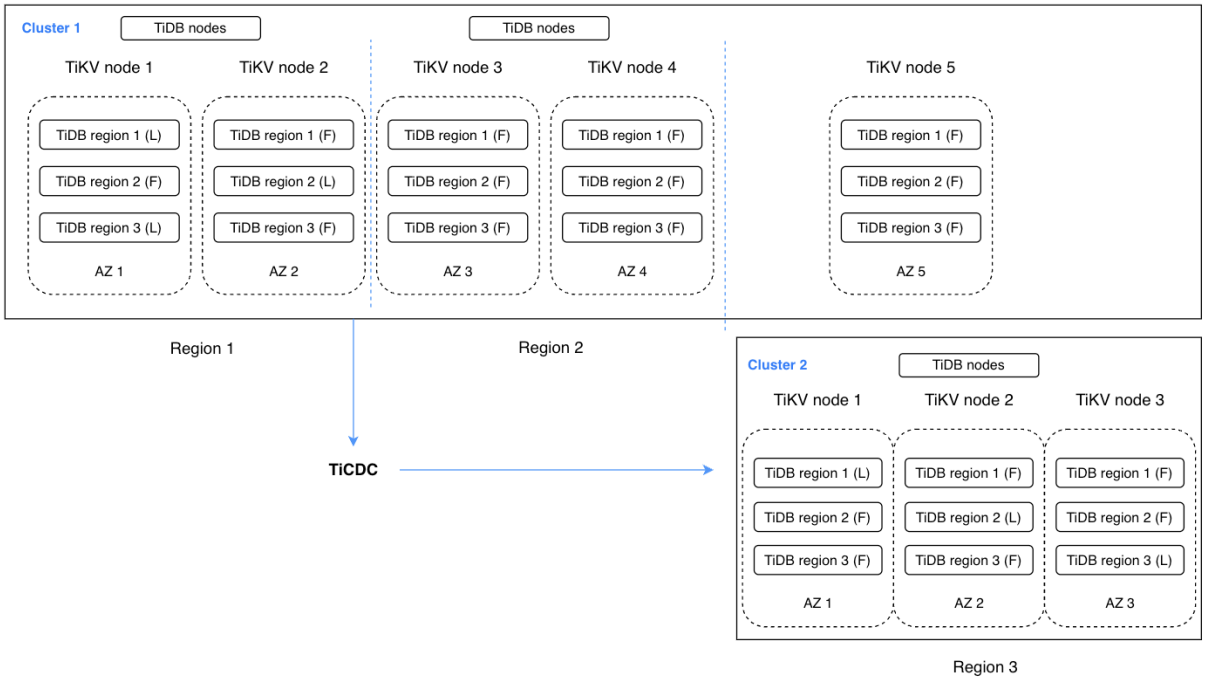
Arsitektur ini sering direkomendasikan oleh banyak vendor database terdistribusi kepada penggunanya sebagai solusi pemulihan bencana. Misalnya, CockroachDB merekomendasikan konfigurasi 3-3-3 untuk mencapai pemulihan bencana tingkat Wilayah; Spanner menyediakan konfigurasi 2-2-1 untuk penerapan multi-Wilayah. Namun, solusi ini tidak menjamin ketersediaan tinggi ketika Wilayah 1 dan 2 tidak tersedia secara bersamaan. Setelah Wilayah 1 benar-benar mati, jika ada node penyimpanan di Wilayah 2 yang bermasalah, sistem mungkin mengalami penurunan kinerja dan bahkan kehilangan data. Jika Anda memerlukan FTT tingkat multi-Wilayah, atau waktu respons sistem yang ketat, solusi ini tetap perlu digabungkan dengan teknologi replikasi log transaksi.

DR multi-Wilayah yang ditingkatkan dengan pengambilan data perubahan

TiCDC adalah alat replikasi data tambahan untuk TiDB. Ini mengambil perubahan data pada node TiKV dan menyinkronkannya dengan sistem hilir. TiCDC memiliki arsitektur yang

mirip dengan sistem replikasi log transaksi, namun lebih terukur dan berfungsi baik dengan TiDB dalam skenario pemulihan bencana.

Konfigurasi berikut berisi dua klaster TiDB. Region 1 dan Region 2 bersama-sama membentuk Cluster 1 yang merupakan cluster 5 replika. Wilayah 1 berisi dua replika untuk melayani operasi baca dan tulis, dan Wilayah 2 berisi dua replika untuk failover cepat jika terjadi bencana di Wilayah 1. Wilayah 3 berisi satu replika untuk mencapai kuorum di antara grup Raft. Cluster 2 di Wilayah 3 berfungsi sebagai cluster pemulihan bencana. Ini berisi tiga replika untuk menyediakan failover cepat jika terjadi bencana di Wilayah 1 dan Wilayah 2. TiCDC menangani sinkronisasi perubahan data antara dua cluster. Arsitektur yang ditingkatkan ini dapat disebut sebagai 2-2-1:1.



Gambar 8. 13 Contoh DR multi-wilayah TiDB dengan TiCDC

Konfigurasi yang tampaknya rumit ini sebenarnya memiliki ketersediaan yang lebih tinggi. Hal ini dapat mencapai target toleransi kesalahan maksimum pada tingkat multi-wilayah, dengan RPO dalam hitungan detik dan RTO dalam hitungan menit. Untuk satu Wilayah, RPO adalah 0 jika tidak tersedia sepenuhnya. Solusi ini menggabungkan keunggulan database terdistribusi dengan pengalaman pemulihan bencana database tradisional. Dan lebih pragmatis Dibandingkan dengan solusi pemulihan bencana berbasis banyak replika dari CockroahDB atau Spanner, solusi ini lebih praktis.

8.4 Toleransi kesalahan dalam sistem terdistribusi

Di dunia yang terhubung saat ini, sistem terdistribusi ada dimana-mana. Mereka membantu menjalankan hal-hal seperti komputasi awan dan media sosial, yang kita gunakan setiap hari. Namun sistem ini terkadang bisa gagal, jadi memastikan sistem berfungsi dengan baik sangatlah penting. Itu sebabnya kita memerlukan toleransi kesalahan. Dalam artikel ini, kita akan membahas mengapa toleransi kesalahan penting dalam sistem terdistribusi dan membahas berbagai cara untuk mewujudkannya.

Memahami Sistem Terdistribusi

Sebelum kita membahas mengapa toleransi kesalahan penting dalam sistem terdistribusi, mari kita pahami dulu apa itu toleransi kesalahan. Sistem terdistribusi adalah sekelompok komputer terhubung yang bekerja sama untuk mencapai tujuan bersama. Daripada melakukan semuanya di satu komputer seperti pada sistem tradisional, sistem terdistribusi menyebarkan tugas dan informasi ke banyak komputer. Hal ini membantu pertumbuhan yang lebih besar, menyeimbangkan pekerjaan, dan menangani kesalahan, menjadikan sistem terdistribusi berguna untuk banyak hal berbeda.

Pentingnya Toleransi Kesalahan

Toleransi kesalahan adalah kemampuan suatu sistem untuk tetap berfungsi dengan baik ketika terjadi kegagalan. Dalam sistem terdistribusi, kegagalan komponen dapat terjadi karena berbagai alasan, seperti kegagalan fungsi perangkat keras, masalah jaringan, atau kesalahan perangkat lunak.

Tanpa adanya mekanisme toleransi kesalahan, satu titik kegagalan dapat menyebabkan gangguan sistem, kehilangan data, dan pengalaman pengguna yang buruk. Pentingnya toleransi kesalahan dalam sistem terdistribusi dapat diringkas dalam beberapa poin penting:

1. Peningkatan Keandalan

Toleransi kesalahan membuat sistem terdistribusi lebih andal. Ini mengurangi dampak masalah sehingga pengguna dapat terus menggunakan sistem tanpa henti. Ketergantungan ini sangat penting untuk bidang-bidang penting seperti perbankan, kesehatan, dan pengelolaan sistem besar.

2. Ketersediaan Tinggi

Toleransi kesalahan menjamin bahwa sistem terdistribusi tetap berjalan, tidak peduli hambatan apa pun yang menghadang! Ketersediaan tinggi sangat penting untuk aplikasi yang tidak mampu menghadapi downtime, seperti situs web e-commerce, platform streaming, dan alat komunikasi - semuanya harus dapat diakses 24/7!

3. Integritas Data

Dalam sistem terdistribusi, data disalin ke banyak tempat untuk kecepatan dan pencadangan yang lebih baik. Metode toleransi kesalahan menjaga data tetap benar dan konsisten, bahkan ketika ada bagian yang rusak atau data berpindah antar tempat.

4. Skalabilitas

Salah satu hal terbaik tentang sistem terdistribusi adalah skalabilitasnya, mereka dapat dengan mudah beradaptasi untuk menangani lebih banyak pekerjaan sesuai kebutuhan. Toleransi kesalahan sangat penting dalam menjaga skalabilitas sistem ini, karena penambahan atau penghapusan node tidak boleh mengganggu operasi sistem secara keseluruhan.

5. Pemulihan Bencana

Dalam sistem terdistribusi, seluruh pusat data mungkin gagal. Strategi toleransi kesalahan, seperti redundansi geografis, dapat membantu skenario pemulihan bencana, memastikan bahwa sistem dapat pulih dan terus beroperasi di lokasi yang berbeda.

Mencapai Toleransi Kesalahan dalam Sistem Terdistribusi

Untuk mencapai toleransi kesalahan dalam sistem terdistribusi, berbagai strategi dan teknik digunakan. Berikut adalah beberapa pendekatan yang paling umum:

1. Redundansi

Redundansi melibatkan replikasi data atau layanan di beberapa node atau komponen. Jika satu node gagal, node lain yang memiliki data dapat mengambil alih tanpa hambatan, memastikan layanan tidak terganggu. Redundansi dapat diterapkan di berbagai tingkatan, termasuk redundansi data, redundansi node, dan redundansi komponen.

2. Penyeimbangan Beban

Penyeimbangan beban adalah teknik yang mendistribusikan lalu lintas masuk atau permintaan secara merata ke beberapa node. Hal ini tidak hanya meningkatkan kinerja tetapi juga meningkatkan toleransi kesalahan. Jika satu node kewalahan atau gagal, penyeimbang beban dapat mengalihkan lalu lintas ke node yang sehat, mencegah kelebihan beban dan waktu henti.

3. Kegagalan dan Kegagalan Kembali

Mekanisme failover dan failback secara otomatis beralih dari komponen yang gagal ke komponen cadangan atau sekunder. Pendekatan ini sering digunakan untuk sistem penting seperti database dan server web. Setelah komponen utama pulih, mekanisme failback beralih kembali ke komponen asli.

4. Replikasi

Replikasi data adalah teknik penting untuk toleransi kesalahan. Dengan mereplikasi data di beberapa node, sistem terdistribusi dapat memastikan ketersediaan data meskipun beberapa node gagal. Berbagai strategi replikasi, termasuk pendekatan master-slave, leader-follower, dan berbasis kuorum, digunakan untuk menjaga konsistensi dan ketersediaan data.

5. Redundansi Geografis

Untuk pemulihan bencana dan ketersediaan tinggi, redundansi geografis digunakan. Hal ini melibatkan replikasi data dan layanan di beberapa pusat data atau lokasi, seringkali di wilayah atau negara berbeda. Jika satu lokasi mengalami kegagalan, sistem dapat terus beroperasi dari lokasi lain.

6. Deteksi dan Pemulihan Kesalahan

Menerapkan mekanisme untuk deteksi kesalahan dan pemulihan sangat penting untuk toleransi kesalahan. Sistem dapat menggunakan teknik seperti detak jantung , pemeriksaan kesehatan, dan prosedur pemulihan otomatis untuk mengidentifikasi dan mengurangi kegagalan secara real-time.

7. Algoritma Konsensus Terdistribusi

Algoritme konsensus terdistribusi seperti Paxos dan Raft memainkan peran penting dalam menjaga konsistensi data dan toleransi kesalahan. Algoritme ini membantu sistem terdistribusi menyetujui urutan operasi dan memastikan bahwa data tetap akurat, bahkan ketika ada partisi jaringan atau kegagalan node.

8. Pemantauan dan Pencatatan

Pemantauan dan pencatatan yang komprehensif sangat penting untuk mengidentifikasi dan mendiagnosis kegagalan. Logging memungkinkan administrator untuk melacak penyebab masalah sementara alat pemantauan memberikan wawasan real-time mengenai kinerja dan kesehatan sistem.

Tantangan Toleransi Kesalahan

Meskipun toleransi kesalahan sangat penting untuk keandalan sistem terdistribusi, hal ini memiliki tantangan tersendiri:

1. Kompleksitas : Penerapan mekanisme toleransi kesalahan dapat secara signifikan meningkatkan kompleksitas sistem terdistribusi, sehingga lebih menantang untuk merancang, menerapkan, dan memelihara.
2. Resource Overhead : Redundansi, replikasi, dan strategi toleransi kesalahan lainnya biasanya memerlukan perangkat keras tambahan dan sumber daya komputasi, yang dapat meningkatkan biaya operasional.
3. Konsistensi vs. Ketersediaan : Menjaga keseimbangan antara konsistensi data dan ketersediaan sistem adalah tantangan umum dalam sistem terdistribusi. Memastikan keduanya bisa jadi rumit, terutama jika ada partisi jaringan.
4. Latensi : Beberapa mekanisme toleransi kesalahan, seperti redundansi geografis, dapat menimbulkan latensi, yang mungkin tidak dapat diterima untuk aplikasi real-time atau latensi rendah.

Bab 9 Big Data dan Sistem Basis Data

9.1 Pengantar ke Big Data

Apa itu Big Data ? saat ini, teknologi sangat berkembang pesat dan masih terus berkembang. Big Data termasuk salah satu dari teknologi yang masih terus berkembang saat ini. Kita akan masuk kepada apa itu Big Data? merupakan sebuah teknologi yang sangat populer dikalangan seorang pengembang suatu aplikasi. Hal yang terpenting untuk menyimpan suatu data adalah basis data. Basis data sangat dibutuhkan sebagai wadah penyimpanan, mengelola informasi dan mengelola sebuah informasi yang berbentuk data secara terstruktur. Contohnya seperti perusahaan besar yang mempunyai banyak data dan membutuhkan suatu tempat penyimpanan yang sangat besar untuk menyimpan sebuah informasi atau data yang ada di perusahaan tersebut.

Sedangkan, untuk sebuah proyek dengan skala kecil, cukup hanya dengan menggunakan perangkat lunak yang bersifat sumber yang terbuka seperti MariaDB, MySQL, Google Cloud SQL, dan lain-lain. Tetapi, jika kita membutuhkan perangkat lunak yang berfungsi untuk menyimpan data yang sangat besar, maka mengakibatkan terjadinya pengolahan sebuah data akan menjadi sangat lambat dan belum cukup efektif. Jadi, jika membutuhkan sebuah peyimpanan yang sangat besar anda bisa menggunakan suatu big data.

Apa itu BIG DATA

Big data adalah sekumpulan data yang memiliki volume yang sangat besar, dan memiliki berbagai data yang terstruktur maupun yang tidak terstruktur dan digunakan untuk membantu sebagai wadah penyimpanan di suatu perusahaan.

Big data dibagi menjadi 5 bagian, yang biasa disebut dengan 5V:

1. Volume (Kapasitas Data)

Volume merupakan suatu kapasitas yang terdapat dalam database. Data base yang memiliki kapasitas data yang sangat besar dan mungkin tidak terbatas tersebut yang dimiliki oleh Big Data. Jadi, kita bisa mencobanya untuk melakukan proses data dengan kapasitas yang sangat besar.

2. Velocity (Kecepatan)

Dalam melakukan transfer data, kecepatan mengirim data tentunya sangat berpengaruh dalam proses pengiriman data. Jadi, jika kita sedang melakukan transfer data, big data memiliki kecepatan transfer data yang memungkinkan kita untuk mentransfer data secara langsung dan tentunya stabil.

3. Variety

Suatu tipe data yang memiliki fungsi dan dapat diolah mulai dari data yang tidak terstruktur, maupun data yang terstruktur. Contoh dari variety adalah data yang berbentuk tulisan biodata sesorang, video, atau bahkan foto tersebut merupakan data yang tidak terstruktur dan harus diproses dalam suatu data base.

4. Value

Value merupakan suatu unsur pada big data yang merupakan sekumpulan data yang belum tetap atau belum teratur pada beberapa kondisi. Hal itu mungkin terjadi pada suatu permasalahan dimana terdapat pembengkakan suatu data yang sangat besar sehingga memproses suatu data yang besar tersebut akan memproses data dengan ukuran suatu penyimpanan yang sangat besar.

5. Veracity

Veracity merupakan bentuk suatu data yang dipercaya kebenarannya, berkualitas, dan dapat diakses dengan mudah. Jadi, mengacu pada kualitas data tersebut, perlu adanya proses untuk menghubungkan dan mengkorelasikan beberapa hubungan data.

Contoh pada veracity adalah data dalam suatu perusahaan. Data yang dimiliki oleh suatu perusahaan yang mempunyai ukuran yang data yang sangat besar akan sulit untuk di susun secara teratur untuk mendapatkan data yang akurat. Jadi, data-data tersebut harus akurat karena digunakan untuk menganalisa data.

Fungsi Big Data

Big data juga memiliki macam-macam fungsi yang sangat penting dalam memproses pengembangan suatu perangkat lunak. Berikut ini terdapat macam-macam fungsi yang terkait dengan big data:

1. Dapat mengetahui pemicu suatu masalah dan kegagalan secara langsung. Mengetahui suatu pemicu atau kegagalan yang terjadi di dalam system database, dan dapat memudahkan untuk menganalisis secara langsung pemicu dari kegagalan tersebut dan dapat meredam terjadinya suatu kegagalan dalam proses penyimpanan data.

2. Dapat mengetahui terjadinya suatu anomali data yang menyimpang yang dialami perusahaan anda. Jadi, Big Data mampu mendeteksi suatu anomali dengan sangat akurat dan tentunya sangat cepat untuk membantu suatu perusahaan anda.

3. Big Data juga sanggup menurunkan biaya, waktu, dan menaikan performa suatu aplikasi. Menggunakan sistem big data mampu mengurangi biaya dalam hal penyimpanan yang sangat dibutuhkan oleh suatu perusahaan. Kemudian, big data mampu mengurangi waktu dalam mengelola suatu data yang ada suatu di perusahaan dan mampu mentransfer data dengan sangat cepat dan akurat. Jadi, Big Data sangat dibutuhkan di perusahaan yang sangat besar.

Big data adalah sekumpulan data yang memiliki volume yang sangat besar, dan memiliki berbagai data yang terstruktur maupun yang tidak terstruktur dan digunakan untuk membantu sebagai wadah penyimpanan di suatu perusahaan. Terdapat 5 unsur dalam big data yaitu volume (kapasitas), velocity (kecepatan), variety (variasi), dan veracity (kebenaran). Dan terdapat macam-macam fungsi dari Big Data.

9.2 Tantangan Penyimpanan dan Analisis Big Data

Pertumbuhan penggunaan teknologi informasi dan komunikasi telah mengubah cara berkomunikasi, terutama dalam sistem penyimpanan data dan penyebaran informasi. Sistem penyimpanan adalah rangkaian teknologi dan proses yang digunakan untuk meningkatkan, mengoptimalkan kinerja sumber daya berdasarkan jumlah penyimpanan yang digunakan. Umumnya mengacu pada penyimpanan data komputer, yang mungkin termasuk memori. Terdapat beberapa kategori yang cukup luas yang dapat mencakup virtualisasi, replikasi, mirroring, keamanan, penyedia hosting dan lainnya. Disini kita akan membahas tantangan penerapan sistem penyimpanan pada big data.

Klasifikasi Data Apa Saja Yang Ada Pada Big Data

Teknologi Big Data mampu menangani banyak jenis data. Secara umum ada 2 kelompok data
yang harus dikelola, yaitu:

1. Data Terstruktur

Sekelompok data dengan tipe, format, dan struktur data yang telah ditentukan. Data terstruktur yang terorganisir dengan baik dapat mudah diakses oleh seseorang atau program

komputer. Fitur yang paling menarik dari data terstruktur adalah mudah dipahami dalam bahasa mesin dan dapat dimanipulasi dalam berbagai cara. Sumber data dapat berupa data transaksional, data OLAP, RDBMS tradisional, file CSV, spreadsheet.

2. Data Tidak Terstruktur

Data tidak terstruktur adalah data yang tidak dapat diolah atau dianalisis dengan alat dan metode konvensional. Sulit untuk merubahnya karena tidak memiliki model yang telah ditentukan, sehingga data harus disimpan dalam format aslinya. Model data merupakan gabungan dari tipe data seperti teks, gambar, video, audio, dan teks.

Gambar 9. 1 Contoh Big Data

Jumlah data berkembang sangat cepat. Pada tahun 2020, setiap manusia di bumi diharapkan menghasilkan 1,7 M/b data per detik. Teknologi Big Data sedang terjadi di Indonesia, penggunaan big data telah mencapai 90% di berbagai belahan dunia. Hanya saja di Indonesia penggunaan dan penerapannya masih sangat terbatas dalam berbagai aspek kehidupan masyarakat. Untuk saat ini, data masih didominasi oleh konsumen internet. Namun, diharapkan ke depan bergerak tepat ke sektor konsumen dan diikuti oleh pemerintah berbasis internet.

Apa Saja Tantangan Penerapan Sistem Penyimpanan Pada Big Data

Tantangan utama dalam pemanfaatan Big Data antara lain eksplorasi. Big Data merupakan proses yang kompleks, belum adanya tata kelola yang jelas dalam pemanfaatan Big

Data. Keterbatasan waktu, biaya yang mahal dan masih jarang perangkat lunak yang dapat digunakan untuk menganalisa data dalam bahasa Indonesia. Serta SDM untuk pengembangan Big Data. Jadi tantangan yang ada saat ini lebih pada bagaimana perusahaan mengoptimalkan data yang dimiliki, untuk memberikan dampak yang positif. Hal yang dapat di uji dalam big data merupakan adalah penyimpanan data yang cukup besar. Keahlian big data dimana setiap pengambilan data dalam bentuk aslinya tanpa menghilangkan salah satu dari data tersebut. Masalah dalam big data memperhitungkan baik kegagalan sistem internal, eksternal, dan kerentanan. Dengan skala data yang cukup besar. Data probabilitas kehilangan beberapa data selama pengambilan data tersebut bisa terjadi dan bisa sangat tinggi kemungkinannya.

9.3 Solusi Basis Data untuk Big Data

Solusi basis data untuk Big Data merupakan elemen kunci dalam menangani volume, kecepatan, dan variasi data yang besar dan beragam. Dalam lingkungan di mana data terus meningkat dengan cepat, organisasi membutuhkan solusi yang dapat mengakomodasi skala yang besar dan menyediakan kinerja yang cepat. Salah satu pendekatan yang banyak digunakan adalah penggunaan basis data NoSQL. Basis data NoSQL, seperti MongoDB, Cassandra, atau HBase, dirancang khusus untuk menangani volume data yang besar dan dapat dengan mudah di-scaling secara horizontal. Mereka memungkinkan penyimpanan data yang terdistribusi dan fleksibel, serta memungkinkan pemrosesan data secara paralel untuk meningkatkan kinerja dan responsifitas. Selain itu, basis data NoSQL juga lebih toleran terhadap kesalahan dan memungkinkan fleksibilitas skema yang memungkinkan penyimpanan data semi-terstruktur atau tidak terstruktur.

Selain basis data NoSQL, teknologi lain yang sering digunakan dalam solusi basis data untuk Big Data adalah teknologi Hadoop. Hadoop adalah kerangka kerja sumber terbuka yang memungkinkan penyimpanan dan pemrosesan data yang sangat besar secara terdistribusi. Salah satu komponen utama dari ekosistem Hadoop adalah Hadoop Distributed File System (HDFS), yang dirancang untuk menyimpan dan mengelola data secara terdistribusi di sejumlah node server. Selain itu, Hadoop juga menyediakan kerangka kerja pemrosesan data yang sangat paralel dengan menggunakan model pemrograman MapReduce. Dengan kombinasi HDFS dan MapReduce, Hadoop memungkinkan organisasi untuk melakukan analisis data yang mendalam dan memproses data dalam skala yang sangat besar dengan kinerja yang baik.

Selain basis data NoSQL dan Hadoop, ada juga solusi basis data lain yang dirancang khusus untuk Big Data, seperti Apache Cassandra, Apache HBase, dan Amazon DynamoDB. Masing-masing solusi ini memiliki keunggulan dan kelemahan tersendiri tergantung pada kebutuhan spesifik organisasi dan karakteristik data yang dihadapi. Namun, pada intinya, solusi basis data untuk Big Data bertujuan untuk menyediakan kemampuan untuk menyimpan, mengelola, dan menganalisis data dalam skala yang sangat besar dengan kinerja yang cepat dan

responsif. Dengan memilih solusi yang tepat dan menerapkan arsitektur yang sesuai, organisasi dapat memanfaatkan potensi nilai dari Big Data mereka dan membuat keputusan yang lebih baik berdasarkan wawasan yang diperoleh.

9.4 Integrasi Big Data dengan Infrastruktur IT

Integrasi Big Data dengan infrastruktur IT yang ada merupakan langkah penting dalam memastikan keberhasilan implementasi solusi Big Data di suatu organisasi. Salah satu aspek kunci dalam integrasi ini adalah memastikan bahwa sistem basis data yang digunakan untuk Big Data dapat berinteraksi secara harmonis dengan infrastruktur IT yang sudah ada. Hal ini memungkinkan organisasi untuk memanfaatkan investasi yang sudah ada dalam infrastruktur, seperti perangkat keras, perangkat lunak, dan keahlian SDM, sambil memperluas kemampuan mereka untuk mengelola dan menganalisis data dalam skala yang lebih besar dan lebih kompleks.

Integrasi Big Data dengan infrastruktur IT tradisional sering kali melibatkan sejumlah tantangan teknis dan organisasi. Salah satu tantangan utama adalah memastikan bahwa sistem basis data Big Data dapat berintegrasi dengan baik dengan infrastruktur perusahaan yang sudah ada, termasuk basis data relasional, aplikasi bisnis, dan alat analisis data. Ini membutuhkan kerja sama yang erat antara tim pengembangan, tim infrastruktur, dan tim bisnis untuk memahami kebutuhan, mengidentifikasi persyaratan integrasi, dan merancang solusi yang sesuai.

Selain itu, integrasi Big Data juga memerlukan penyesuaian dalam hal keamanan, manajemen data, dan pemantauan performa. Misalnya, perlu memastikan bahwa data yang disimpan dan diakses melalui sistem basis data Big Data tetap terlindungi dan mematuhi kebijakan keamanan yang berlaku di perusahaan. Selain itu, penting juga untuk memiliki alat pemantauan yang efektif yang memungkinkan organisasi untuk memantau kinerja sistem basis data Big Data secara terus-menerus dan mengidentifikasi masalah atau bottle neck dengan cepat.

Integrasi Big Data dengan infrastruktur IT yang sudah ada juga dapat memberikan manfaat besar dalam hal pengelolaan data yang lebih holistik dan pemahaman yang lebih mendalam tentang operasi bisnis. Dengan menggabungkan data dari berbagai sumber, termasuk sistem basis data tradisional dan data Big Data, organisasi dapat memperoleh wawasan yang lebih lengkap tentang kinerja operasional, perilaku pelanggan, dan tren pasar yang dapat digunakan untuk menginformasikan pengambilan keputusan bisnis yang lebih baik.

Pada akhirnya, integrasi Big Data dengan infrastruktur IT merupakan investasi jangka panjang yang dapat membantu organisasi untuk tetap relevan dan bersaing di era digital yang terus berubah. Dengan merencanakan dengan cermat, berkolaborasi secara efektif antara berbagai tim dan departemen, dan memilih solusi basis data yang sesuai, organisasi dapat memanfaatkan potensi nilai dari Big Data mereka dan mencapai keunggulan kompetitif yang berkelanjutan

Bab 10 Tantangan Masa Depan dalam Sistem Basis Data

10.1 Perkembangan Teknologi Basis Data

Basis data masa depan akan lebih dari sekedar basis data . Sebaliknya, ini akan menjadi platform yang berfokus pada pengalaman pengguna dan mengabstraksi teknologi. Melihat evolusi pengembangan aplikasi, sejumlah besar upaya telah mempermudah pembuatan dan penerapan aplikasi. Demikian pula, basis data perlu berevolusi untuk kemudahan penggunaan dengan pengalaman pengembang yang menyenangkan dan antarmuka pengguna yang patut dicontoh yang menampilkan fitur dan fungsi yang benar-benar akan digunakan oleh para insinyur.

Berikut tujuh fitur utama yang diperlukan untuk memetakan perkembangan database masa depan.

1. Tanpa Server secara Default

Tanpa server tidak diragukan lagi merupakan perubahan paradigma paling signifikan dalam rekayasa perangkat lunak dalam lima tahun terakhir dan yang paling sedikit direalisasikan. Saat ini, tidak banyak orang yang sepenuhnya menggunakan serverless. Ini adalah peralihan dari membeli server ke layanan yang melakukan apa yang Anda butuhkan dan berkembang seiring pertumbuhan Anda.

Tanpa server secara default adalah satu-satunya cara untuk memastikan database masa depan akan mudah dioperasikan dan dapat diandalkan secara konsisten. Tanpa server berkembang dari menjalankan fungsi di cloud menjadi standar bagi tim pengembangan yang bergerak cepat yang tidak ingin khawatir tentang vCPU atau kumpulan node tetapi fokus pada hasil.

Perusahaan-perusahaan yang melakukan hal tersebut dengan baik masih menggunakan server, namun mereka memanfaatkan produk yang dapat diskalakan secara independen, berapa pun jumlah server yang mereka miliki. Vendor komponennya dapat meningkatkan jumlah sumber daya yang dialokasikan kepada mereka, dan mereka hanya membayar sesuai penggunaan, dan arsitektur solusinya dapat diskalakan dengan cara tersebut. Mereka tidak membeli satu server pun dan mendorongnya sekuat tenaga. Mereka memberi tahu satu platform, 'ini kode saya,' mereka memberi tahu platform lain, 'ini data saya,' dan kemudian skalanya ditingkatkan berdasarkan penggunaan, jauh lebih jauh dibandingkan membeli satu server.

Ketika orang berpikir tanpa server, mereka berpikir tentang JavaScript dan Lambda, tetapi ada juga PHP tanpa server di luar sana. Bukan hanya komunitas JavaScript. Paradigma perangkat lunak lain juga mendukung hal ini. Kita akan melihat semakin banyak orang melakukan hal ini

ketika mereka perlu melakukan beberapa kode cepat dan mereka ingin mengirimkan ke produksi secepat mungkin dan menskalakannya secepat mungkin.

2. Mudah Dioperasikan

Kemajuan paling signifikan dalam DevOps telah membebaskan pengembang dari operasi kompleks yang membebani operasi mereka sehari-hari. Tim teknik saat ini diatur berdasarkan tujuan produk dan bisnis, bukan pemeliharaan dan servis. Namun database belum bisa mengejar ketinggalan. Tim produk masih mengandalkan para ahli untuk melakukan tugas operasi database yang terlalu rumit.

Basis data masa depan harus sesuai dengan alur kerja DevOps setiap orang. Ini harus dapat digunakan oleh pengembang mana pun dan dimanfaatkan oleh setiap tim dalam organisasi teknis. Basis data harus dapat dioperasikan oleh semua orang dan tidak lagi memerlukan admin basis data untuk mengelola pencadangan dan perencanaan kapasitas atau mengkhawatirkan pemulihan bencana.

Fitur-fitur yang berpusat pada pengembang seperti percabangan basis data, lingkungan pengembang basis data, penskalaan otomatis, dan penagihan berbasis penggunaan akan menjadi hal yang lumrah.

3. Dapat diandalkan

Dapat diandalkan berarti lebih sulit untuk mencapai batas penskalaan. Hal ini tidak hanya berarti bahwa database sudah ada, namun Anda juga mendapatkan informasi yang benar ketika terjadi masalah, dan Anda dapat secara kolaboratif membantu meningkatkan skala.

Tidak ada yang berskala tanpa batas. Tidak ada cukup komputer di dunia untuk mengukur sesuatu hingga tak terbatas. Anda akan selalu dapat menanyakan database dan mengambil terlalu banyak informasi darinya dengan terlalu cepat. Tidak peduli berapa banyak mobil yang Anda miliki atau berapa banyak tenaga kuda yang Anda miliki, Anda tidak dapat melaju sejuta mil per jam. Basis data yang baik harus memberi tahu Anda ketika Anda mencapai batasan tersebut dan membantu Anda melewatinya, baik dalam skala kecil, menengah, atau skala besar.

4. Dapat dioperasikan

Basis data masa depan akan dapat diakses oleh setiap klien – mulai dari lemari es pintar hingga armada kendaraan pengiriman otonom. Namun pendekatan agnostik klien ini memerlukan perencanaan untuk lebih banyak transaksi dan koneksi daripada yang dapat dikelola sebagian besar database saat ini secara bersamaan.

Di masa depan, database dapat menangani koneksi tak terbatas melalui pengumpulan koneksi cerdas, replikasi kueri, dan perutean kueri cerdas. Basis data juga perlu melayani beragam koneksi dan klien. Meskipun kita sudah melihat banyak database yang menawarkan API web dan tipe koneksi baru, seiring dengan pengujiannya dalam skala besar, kita akan mulai melihat database tersebut diadopsi dan digunakan untuk mendukung aplikasi generasi berikutnya.

5. Didistribusikan Secara Global, Tersedia Secara Lokal

Saat ini, sebagian besar bisnis dan aplikasi tidak memerlukan database global, yang sering kali menimbulkan kerumitan dan biaya pengoperasian yang tidak perlu. Namun, ada banyak alasan mengapa database masa depan perlu didistribusikan secara global, karena database tersebut harus ditempatkan di tempat yang paling Anda butuhkan.

Kuncinya adalah mengetahui apa kebutuhan Anda. Banyak orang melihat multi-wilayah dan menganggapnya sebagai pembacaan yang lebih cepat, failover, penskalaan, dan melebihi kecepatan cahaya. Pada akhirnya, itu semua adalah serangkaian pengorbanan. Misalnya, saat Anda membaca dari replika lokal, Anda mengorbankan konsistensi karena Anda memiliki replikasi asinkron. Namun hal ini benar-benar memanfaatkan internet global untuk memberikan hasil yang lebih baik jika diperlukan.

Kesiapan kepatuhan dengan data yang disimpan secara regional memungkinkan Anda melakukan penskalaan dengan cepat ke wilayah baru dan memberikan pengalaman pengguna yang luar biasa dengan menyajikan data yang tersedia secara lokal dengan cepat. Aplikasi bertransaksi tinggi dan berlatensi rendah seperti game seluler sudah membutuhkan kemampuan ini.

Namun ada satu peringatan: Distribusi data global tidak boleh melanggar prinsip nomor dua, yaitu mudah dioperasikan. Alat harus elegan dan memiliki antarmuka yang jelas karena tujuan utamanya adalah membuat pengoperasian lebih mudah, bukan lebih sulit.

6. Cerdas

Basis data masa depan dimaksudkan untuk digunakan oleh masyarakat dan akan dapat memberi tahu mereka bagaimana basis data tersebut perlu digunakan dan bagaimana menjadikannya lebih baik. Semua alat yang Anda beli hari ini hanya dapat memberi Anda data yang Anda perlukan untuk membuat beberapa tebakan. Basis data di masa depan akan menggantikannya dengan rekomendasi konkrit, kemudian tindakan penyembuhan diri.

Hal ini dimulai dengan pengumpulan dan observasi data dasar, lalu berlanjut ke, “mari kita catat setiap kueri yang dibuat seseorang dan gunakan beberapa mesin keputusan dan beberapa AI, lalu rekomendasikan bagaimana mereka dapat melakukan sharding secara

otomatis.” Ia akan dapat mengatakan, “Saya telah mencatat penggunaan database Anda selama 30 hari terakhir; ini adalah bagaimana Anda harus melakukan shard untuk kinerja optimal. Klik tombolnya, dan kami akan melakukannya.”

Basis data masa depan akan mengatakan, “Anda melakukan penskalaan terlalu jauh untuk satu mesin pada tingkat per pengguna. Anda harus membaginya menjadi dua atau membaginya menjadi empat, dan kami menemukan cara melakukannya untuk Anda, jadi Anda tidak perlu memikirkannya. Kami telah membagi data Anda atas nama Anda”.

7. Dapat Diskalakan Secara Linear dan Horizontal

Basis data di masa depan juga perlu dirancang untuk skala masa depan. Hanya segelintir perusahaan di dunia yang telah mencapai skala yang menguji batas-batas teknologi saat ini, namun seiring dengan berkembangnya informasi, semakin banyak perusahaan yang akan mencapainya. Basis data berikutnya akan melakukannya dengan melakukan penskalaan secara otomatis baik secara vertikal maupun horizontal dalam hitungan detik. Banyak database yang ada memerlukan masa pemeliharaan, replikasi, dan failover sebelum dapat ditingkatkan atau diperkecil skalanya. Namun downtime tidak dapat diterima, dan skala tidak boleh mengorbankan kinerja. Dan itu tidak harus terjadi. Basis data masa depan akan diskalakan secara cerdas dan tanpa kesalahan pada halaman status.

Singkatnya, konsumen harus bisa mendapatkan apa yang mereka butuhkan saat mereka membutuhkannya, tanpa risiko. Seharusnya mudah untuk mengatakan, “Saya menggandakan jumlah server, saya menggandakan jumlah throughput. Saya telah melipatgandakan jumlah server dan jumlah throughput.” Inilah yang disediakan oleh sharding. Sharding yang baik terjadi secara otomatis dalam skala super tinggi, dan Anda hanya akan membayar sesuai penggunaan Anda seiring berjalannya waktu. Basis data masa depan akan melakukan semua ini tanpa pengguna harus memutar kenop.

10.2 Big Data dan Analisis Prediktif

Big Data telah menjadi kekuatan revolusioner dalam era digital saat ini. Kemampuannya untuk mengumpulkan, menyimpan, dan menganalisis volume data yang besar dan beragam telah membuka pintu untuk pemahaman yang lebih dalam terhadap tren, pola, dan perilaku di berbagai bidang. Salah satu kekuatan utama Big Data terletak pada kemampuannya untuk memprediksi.

1. Analisis yang Mendalam Berkat Big Data

Big Data memungkinkan organisasi untuk menggali wawasan yang tak terbatas dari data yang dikumpulkan. Dengan algoritma dan teknik analisis yang canggih, data yang dikumpulkan

dari berbagai sumber seperti sensor, media sosial, transaksi, dan lainnya dapat diproses secara efisien. Kemampuan untuk menganalisis data dalam skala besar memungkinkan deteksi pola yang tidak dapat terlihat dengan analisis konvensional.

2. Prediksi Tren dan Perilaku

Kekuatan Big Data terletak pada kemampuannya untuk memprediksi tren masa depan. Dengan memanfaatkan data historis dan saat ini, model prediktif yang ditenagai oleh Big Data dapat mengidentifikasi pola dan perilaku yang muncul. Contohnya adalah dalam industri ritel di mana data transaksi pelanggan dapat memberikan pemahaman yang dalam tentang preferensi konsumen, membantu dalam meramalkan permintaan, dan mengantisipasi tren belanja.

3. Penerapan di Berbagai Bidang

Big Data telah membuka peluang tak terbatas dalam berbagai sektor. Dalam dunia kesehatan, analisis Big Data memungkinkan prediksi penyebaran penyakit, pengembangan obat-obatan, dan personalisasi perawatan medis. Di sektor keuangan, analisis Big Data digunakan untuk mendeteksi kecurangan, meramalkan pergerakan pasar, dan mengoptimalkan investasi.

4. Tantangan dan Potensi Masa Depan

Meskipun memiliki potensi besar, penggunaan Big Data juga menghadapi tantangan. Perlindungan privasi dan keamanan data, kesesuaian regulasi, serta kemampuan untuk mengelola volume data yang semakin meningkat menjadi beberapa kendala yang perlu diatasi. Namun, dengan inovasi yang terus berkembang, seperti kecerdasan buatan (AI) dan teknologi analisis yang lebih maju, potensi Big Data untuk memprediksi masa depan terus berkembang.

Big Data bukan sekadar tentang jumlah data yang besar, tetapi lebih tentang bagaimana data tersebut diolah, dianalisis, dan dimanfaatkan untuk menghasilkan wawasan yang bermanfaat. Dengan terus mengembangkan teknologi dan strategi analisis, Big Data akan terus menjadi kekuatan besar yang memungkinkan prediksi yang lebih akurat dan mendalam di berbagai sektor kehidupan.

10.3 Kecerdasan Buatan dalam Sistem Basis Data

Kecerdasan Buatan (AI) dalam sistem basis data menjanjikan kemajuan yang luar biasa untuk masa depan, mengubah paradigma dalam cara data dikelola, dianalisis, dan dimanfaatkan oleh organisasi. Salah satu potensi terbesar dari kecerdasan buatan adalah kemampuannya untuk meningkatkan efisiensi operasional dan pengambilan keputusan dengan memanfaatkan potensi besar dari data yang tersedia. Di masa depan, AI dapat digunakan dalam berbagai cara yang mengubah permainan dalam konteks sistem basis data.

Salah satu aplikasi utama AI dalam sistem basis data adalah dalam analisis prediktif yang lebih canggih. Dengan memanfaatkan teknik machine learning dan deep learning, sistem basis data akan dapat mempelajari pola yang lebih kompleks dalam data dan membuat prediksi yang lebih akurat tentang tren masa depan, permintaan pasar, atau perilaku konsumen. Ini akan memberikan organisasi keunggulan yang signifikan dalam merencanakan strategi bisnis dan mengantisipasi perubahan pasar.

Selain itu, kecerdasan buatan juga dapat digunakan dalam otomatisasi tugas-tugas rutin dalam pengelolaan basis data, seperti pemeliharaan, pemantauan kinerja, dan penyesuaian konfigurasi. Sistem basis data yang dilengkapi dengan AI akan dapat mengidentifikasi masalah secara otomatis, meresponsnya dengan cepat, dan bahkan melakukan tindakan perbaikan sendiri tanpa intervensi manusia. Hal ini akan mengurangi beban kerja administratif dan memungkinkan sumber daya manusia untuk fokus pada tugas-tugas yang lebih strategis dan bernilai tambah.

Selain itu, AI juga akan membawa inovasi dalam antarmuka pengguna sistem basis data. Pengguna akan dapat berinteraksi dengan basis data menggunakan bahasa alami atau antarmuka yang lebih intuitif, seperti asisten virtual atau chatbots, yang memungkinkan pengguna untuk mengakses informasi atau melakukan analisis data dengan lebih cepat dan efisien. Ini akan membuka akses ke data bagi sejumlah orang yang sebelumnya tidak memiliki latar belakang teknis yang kuat, memperluas cakupan penggunaan data dalam organisasi.

Tidak hanya itu, kecerdasan buatan juga akan membawa kemajuan dalam pengelolaan dan analisis data yang tidak terstruktur, seperti teks, gambar, atau video. Teknik seperti natural language processing (NLP) dan computer vision akan memungkinkan sistem basis data untuk mengolah dan menganalisis data ini dengan lebih efektif, membuka pintu untuk wawasan yang lebih dalam dan berharga dari data yang sebelumnya sulit diakses.

Dengan memanfaatkan potensi AI dalam sistem basis data, organisasi akan dapat mengoptimalkan penggunaan data mereka, meningkatkan efisiensi operasional, dan membuat keputusan yang lebih baik berdasarkan informasi yang lebih akurat dan relevan. Namun, sambil mengadopsi teknologi ini, penting untuk memperhatikan aspek etika, keamanan, dan privasi yang terkait dengan penggunaan data, serta memastikan bahwa organisasi memiliki keterampilan dan keahlian yang diperlukan untuk memanfaatkan potensi penuh dari AI dalam sistem basis data.

Analisis Prediktif untuk Personalisasi Layanan

Sebuah perusahaan e-commerce menggunakan teknik machine learning dalam sistem basis datanya untuk menganalisis pola pembelian dan perilaku pelanggan. Berdasarkan data historis tentang preferensi pembelian, lokasi geografis, dan aktivitas online, sistem dapat membuat prediksi yang akurat tentang produk atau layanan yang mungkin diminati oleh setiap pelanggan secara individual. Dengan demikian, perusahaan dapat menyajikan rekomendasi yang dipersonalisasi kepada pelanggan, meningkatkan pengalaman mereka dan meningkatkan konversi penjualan.

Otomatisasi Pemeliharaan dan Pemantauan Kinerja

Sebuah organisasi menggunakan kecerdasan buatan dalam sistem basis data mereka untuk memantau kinerja infrastruktur secara real-time dan mendeteksi anomali yang menunjukkan kemungkinan kegagalan atau masalah kinerja. Dengan menggunakan algoritma machine learning, sistem dapat mengidentifikasi pola perilaku yang abnormal dan memberikan peringatan kepada tim operasi untuk tindakan perbaikan yang tepat waktu. Ini membantu organisasi dalam meminimalkan downtime dan meningkatkan ketersediaan layanan.

Interaksi Pengguna yang Lebih Intuitif

Sebuah platform analisis data menggunakan teknologi natural language processing (NLP) dalam sistem basis datanya untuk memungkinkan pengguna untuk berinteraksi dengan data menggunakan bahasa alami. Pengguna dapat mengajukan pertanyaan kompleks atau meminta analisis data tanpa perlu pengetahuan teknis yang mendalam tentang query SQL atau analisis data. Sistem secara otomatis menganalisis pertanyaan pengguna, mengekstraksi informasi yang relevan dari basis data, dan menyajikan jawaban dalam format yang mudah dimengerti.

Analisis Data Tidak Terstruktur untuk Penemuan Wawasan Baru

Sebuah organisasi kesehatan menggunakan kecerdasan buatan dalam sistem basis data mereka untuk menganalisis data medis tidak terstruktur, seperti catatan medis elektronik, gambar MRI, dan laporan laboratorium. Dengan menggunakan teknologi computer vision dan NLP, sistem dapat mengidentifikasi pola dan informasi yang tersembunyi dalam data medis ini, seperti diagnosa penyakit yang mungkin terlewat atau hubungan antara faktor risiko dan prognosis penyakit. Ini membantu dokter dalam membuat keputusan yang lebih tepat dan efisien dalam perawatan pasien.

Dalam semua contoh di atas, kecerdasan buatan memainkan peran kunci dalam meningkatkan pengelolaan dan analisis data dalam sistem basis data, membuka peluang baru untuk inovasi, efisiensi, dan keunggulan kompetitif di berbagai industri.

10.4 Implikasi Etis dan Hukum

Implikasi etis dan hukum dari perkembangan sistem basis data masa depan menjadi semakin penting seiring dengan peningkatan kompleksitas teknologi dan ketergantungan yang lebih besar pada data dalam kehidupan sehari-hari. Salah satu implikasi utama adalah masalah privasi dan perlindungan data. Dalam sistem basis data yang semakin canggih, volume dan jenis data yang dikumpulkan tentang individu dapat menjadi sangat besar dan mencakup informasi yang sangat sensitif. Ini menimbulkan pertanyaan tentang bagaimana data tersebut digunakan, disimpan, dan diakses, serta siapa yang memiliki kontrol atasnya. Penting untuk memastikan bahwa aturan privasi dan keamanan data yang ketat diterapkan untuk melindungi hak privasi individu dan mencegah penyalahgunaan data.

Selain itu, ada juga pertanyaan tentang bias dan diskriminasi yang mungkin muncul dalam analisis data dan pengambilan keputusan yang didorong oleh kecerdasan buatan dalam sistem basis data. Meskipun AI dapat membantu dalam mengidentifikasi pola dan tren yang tidak terlihat sebelumnya dalam data, ada risiko bahwa model AI dapat memperkuat atau bahkan memperluas bias yang ada dalam data. Hal ini dapat mengakibatkan keputusan yang tidak adil atau diskriminatif terhadap kelompok tertentu, seperti minoritas atau kelompok rentan. Oleh karena itu, penting untuk mempertimbangkan etika pengembangan dan penggunaan teknologi AI, serta memastikan bahwa model AI diuji secara menyeluruh untuk mengidentifikasi dan mengatasi bias yang tidak diinginkan.

Selanjutnya, terdapat juga pertanyaan tentang transparansi dan akuntabilitas dalam penggunaan kecerdasan buatan dalam sistem basis data. Dalam beberapa kasus, model AI mungkin menghasilkan keputusan atau rekomendasi yang sulit dipahami oleh manusia, membuat sulit untuk menjelaskan atau mempertanggungjawabkan keputusan tersebut. Ini menimbulkan kekhawatiran tentang bagaimana keputusan yang didorong oleh AI diputuskan dan bagaimana mereka dapat dipertanggungjawabkan secara etis. Oleh karena itu, penting untuk membangun kerangka kerja yang jelas untuk mengelola dan memonitor penggunaan AI dalam sistem basis data, serta untuk memastikan bahwa proses pengambilan keputusan yang diberdayakan oleh AI tetap transparan dan dapat dipertanggungjawabkan.

Selain itu, ada juga implikasi hukum yang perlu dipertimbangkan dalam pengembangan sistem basis data masa depan. Regulasi data dan privasi, seperti General Data Protection Regulation (GDPR) di Uni Eropa, telah semakin ketat dalam beberapa tahun terakhir, menempatkan tanggung jawab yang lebih besar pada organisasi untuk melindungi data individu dan memastikan kepatuhan dengan peraturan yang berlaku. Oleh karena itu, organisasi perlu

memastikan bahwa sistem basis data mereka mematuhi regulasi yang berlaku dan memiliki prosedur yang tepat untuk mengelola dan melaporkan pelanggaran privasi data.

Keadilan dan Keterbukaan

Penggunaan kecerdasan buatan dalam sistem basis data dapat menimbulkan kekhawatiran tentang kesenjangan digital dan akses yang tidak merata terhadap teknologi. Ada risiko bahwa teknologi AI dapat meningkatkan ketimpangan yang ada dengan menguntungkan kelompok yang sudah beruntung secara ekonomi atau sosial. Oleh karena itu, penting untuk memastikan bahwa pengembangan dan implementasi sistem basis data masa depan didasarkan pada prinsip keadilan dan keterbukaan, serta memastikan bahwa manfaatnya dapat dirasakan secara merata oleh semua anggota masyarakat.

Pendidikan dan Kesadaran

Implikasi etis dan hukum dari sistem basis data masa depan juga menekankan pentingnya pendidikan dan kesadaran tentang penggunaan teknologi data-driven. Organisasi perlu berinvestasi dalam memberikan pelatihan dan pendidikan kepada karyawan mereka tentang etika dan keamanan data, serta membangun kesadaran di antara pengguna tentang risiko dan tanggung jawab yang terkait dengan penggunaan data pribadi dan sensitif.

Kolaborasi dan Regulasi

Penanggulangan implikasi etis dan hukum dari sistem basis data masa depan memerlukan kolaborasi antara pemerintah, industri, dan masyarakat sipil. Regulasi yang tepat diperlukan untuk mengatur penggunaan dan perlindungan data, sementara kerja sama antar sektor dapat membantu mengidentifikasi dan menyelesaikan tantangan yang muncul dengan cepat. Ini mencakup pembentukan standar dan pedoman yang diterima secara luas untuk pengembangan dan penggunaan teknologi AI dalam sistem basis data.

Pengawasan dan Akuntabilitas

Terakhir, penting untuk menetapkan mekanisme pengawasan dan akuntabilitas yang efektif untuk memastikan bahwa sistem basis data masa depan digunakan secara etis dan sesuai dengan hukum. Ini dapat mencakup pembentukan badan pengawas independen, audit rutin terhadap praktik dan kebijakan penggunaan data, serta mekanisme pelaporan dan penanganan pelanggaran yang efisien.

Dengan mempertimbangkan aspek-aspek ini, organisasi dan masyarakat dapat memastikan bahwa perkembangan sistem basis data masa depan memberikan manfaat yang

maksimal sambil meminimalkan risiko dan dampak negatifnya. Ini memungkinkan kita untuk memanfaatkan potensi besar dari teknologi data-driven dalam menciptakan masyarakat yang lebih inklusif, adil, dan berkelanjutan.

Daftar Pustaka

Bakhel Share. (2020, 12, Oktober). Sistem Basis Data (Database Systems): Definisi, Pengertian, Bahasa, Komponen dan Fungsi. Diakses pada 4 Mei 2024, dari https://www.bakhel.com/2020/10/pengertian-definisi-bahasa-komponen-fungsi-sistem-basis-data.html.

Lucidchart. (2023, 8, Januari). Apa itu Model Basis Data. Diakses pada 4 Mei 2024, dari https://www.lucidchart.com/pages/database-diagram/database-models.

Microsoft 365. (2016, 15, Mei). Dasar-dasar desain database. Diakses pada 4 Mei 2024, dari https://support.microsoft.com/id-id/topic/dasar-dasar-desain-database-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#.

Ginfa. (2021,7,Januari). Normalisasi Database: Pengertian, Tujuan dan Cara Melakukannya. Diakses pada 4 Mei 2024, dari **https://www.jojonomic.com/blog/normalisasi-database/**.

Eprints. (2022,10,Januari). IMPLEMENTASI SISTEM. Diakses pada 4 Mei 2024, dari https://eprints.umk.ac.id/4781/6/BAB%2BV.pdf.

WingArc1st. (2016,7,Maret). Pengaturan basis data. Diakses pada 4 mei 2024, dari https://cs.wingarc.com/manual/ia/doc/10.8/id/1603132.html.

Percona. (2023,30,Agustus). Pencadangan dan Pemulihan Basis Data: Yang Harus Anda Ketahui. Diakses pada 4 Mei 2024, dari https://www.percona.com/blog/backup-and-recovery-for-databases-what-you-should-know/.

Repository. (2020 25,April). Sistem Basis Data MANAJEMEN TRANSAKSI. Diakses pada 4 Mei 2024, dari https://repository.unikom.ac.id/54554/1/7%20-%20Sistem%20Basis%20Data%20-%20Manajemen%20Transaksi.pdf.

Suryadi Andri. (2023,4,Februari). Basis Data dan Sistem Basis Data. Diakses pada 4 Mei 2024, dari https://pustaka.ut.ac.id/lib/wp-content/uploads/pdfmk/MSIM4206-M1.pdf.

Daragh Ó Tuama. (2024,1,Mei). Pemrograman Basis Data. Diakses pada 4 Mei 2024, dari https://codeinstitute.net/global/blog/database-programming/.

Kanade Vijay. (2022,11,Juli). Apa itu SQL? Pengertian, Unsur, Contoh, dan Kegunaannya Tahun 2022. Diakses pada 4 Mei 2024, dari https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-sql/.

Repository.(2020,6,Desember). DESAIN BASIS DATA. Diakses pada 4 Mei 2024, dari https://repository.dinus.ac.id/docs/ajar/Dtbs_7_Desain_Basis_Data.pdf.

Imperva. (2023,1,Desember). Keamanan Basis Data. Diakses pada 4 Mei 2024, dari https://www.imperva.com/learn/data-security/database-security/.

Cloudian. (2023,17, Mei). Strategi Perlindungan Data: 10 tujuan Strategi yang Efektif. Diakses pada 4 Mei 2024, dari https://cloudian-com.translate.goog/guides/data-protection/data-protection-strategy-10-components-of-an-effective-strategy/?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc.

Brenda Gratas. (2022,20,Juni). Audit Basis Data: Praktik Terbaik untuk Mengamankan Data Anda. Diakses pada 4 Mei 2024, dari https://blog.invgate.com/database-auditing.

Oracle Docs. (2019,14,Januari). Panduan Administrator Basis Data. Diakse pada 4 Mei 2024, dari https://docs.oracle.com/en/database/oracle/oracle-database/19/admin/distributed-database-concepts.html#GUID-1E67C35A-097D-4BBF-B2A9-F8954A5C603B.

Scylladb. (2023,22,Juli). Replikasi Basis Data. Diakses pada 4 Mei 2024, dari https://www.scylladb.com/glossary/database-replication/.

Gao Allen. (2022,19,Oktober). Pemulihan Bencana untuk Basis Data: Perkembangannya Selama Bertahun-Tahun. Diakses pada 4 Mei 2024, dari https://www.pingcap.com/blog/disaster-recovery-for-databases-how-it-evolves-over-years/.

Sofwan A. Lawal. (2023,14,Oktober). Toleransi kesalahan dalam sistem terdistribusi. Diakses pada 4 Mei 2024, dari https://blog.sofwancoder.com/fault-tolerance-in-distributed-systems.

Universitas Medan Area. (2022,12,Januari). Apa itu Big Data? Pengertian Big Data Sekaligus Beserta Fungsi dari Suatu Big Data. Diakses pada 4 Mei 2024, dari https://pp.uma.ac.id/2022/01/12/apa-itu-big-data-pengertian-big-data-dan-fungsinya/.

Van Wiggerern Nick. (2023,25,Januari). Basis Data Masa Depan: Tujuh Prinsip Utama. Diakses pada 4 Mei 2024, dari https://devops.com/the-database-of-the-future-seven-key-principles/.

Biodata Penulis

Yofhanda Septi Eirlangga, S.Kom., M.Kom. Lahir di Kota Padang provinsi Sumatera Barat pada tanggal 22 September 1996. Dalam menempuh Pendidikan dimulai dari Sekolah Dasar SDN 05 Surabayo tamat tahun 2008, SMPN 3 Lubuk Basung tamat pada tahun 2011 dan di SMAN 2 Lubuk Basung tamat pada tahun 2014. Lalu melanjutkan ke pendidikan tinggi swasta yaitu Sarjana (S1) Universitas Putra Indonesia YPTK Padang, lulus pada tahun 2018 dengan jurusan Sistem Informasi. Kemudian melanjutkan program Pascasarjana (S2) di Universitas Putra Indonesia YPTK Padang dan lulus pada tahun 2019 Program Studi Teknik Informatika Konsentrasi Sistem Informasi. Penulis mengabdikan diri sebagai salah satu Dosen di Bidang Ilmu Komputer khususnya Program Studi Sistem Informasi di Universitas Adzkia dan menjadi Dosen tetap pada maret tahun 2021 di kampus tersebut. Saat ini menjadi bagian Struktural di Universitas Adzkia sebagai Ketua Unit Pelayanan Terpadu (UPT) ICT.

Angga Putra Juledi, S.Kom., M.Kom. Lahir di Kota Padang pada tanggal 19 Juli 1994. Dalam menempuh Pendidikan dimulai dari Sekolah Dasar SDN 19 Padang tamat tahun 2006, SMPN 3 Padang tamat pada tahun 2009, dan di SMA Pertiwi 2 Padang tamat pada tahun 2012. Lalu melanjutkan ke pendidikan perguruan tinggi swasta yaitu S1 (Sarjana) Universitas Putra Indonesia "YPTK" Padang lulus pada tahun 2018 dengan jurusan Sistem Informasi, Dan melanjutkan Program Pascasarjana (S2) di Universitas Putra Indonesia "YPTK" Padang pada tahun 2019 Program Studi Teknik Informatika. Konsentrasi Sistem Informasi. Saya mengabdikan diri sebagai salah satu Dosen di bidang Ilmu Komputer pada Fakultas Sains Dan Teknologi dengan Program Studi Sistem Informasi di Universitas LabuhanBatu dan menjadi dosen tetap pada tahun 2020 pada kampus tersebut. Paat ini menjadi bagian Struktural di Universitas Labuhanbatu sebagai Kepala Bagian Sumber Daya Manusia periode 2023 s/d 2027. Buku pertama terbit pada 31 Desember 2021 dengan judul ***Internetworking Dan TCP/IP.*** Buku kedua terbit pada tanggal 17 Oktober 2023 dengan judul **Panduan Belajar HTML, CSS, dan JavaScript untuk Pemula**. Ini merupakan buku ke 3 yang penulis tulis.

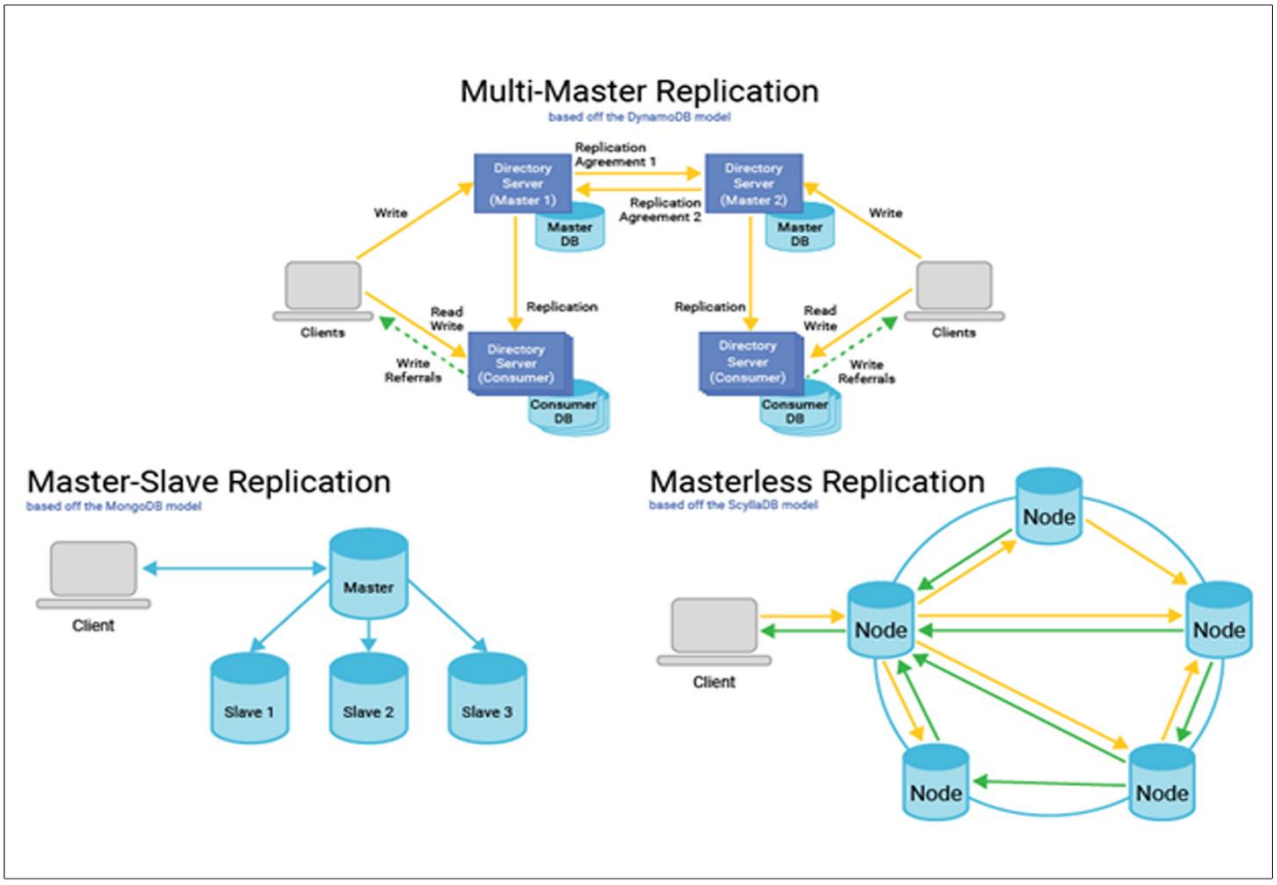
Kiki Hariani Manurung S.Kom, M.Kom., merupakan salah satu Dosen tetap sejak maret 2023 yang mengabdikan diri di Universitas Adzkia, Padang, Sumatera Barat. Merupakan salah satu dosen Ilmu Komputer pada Program Studi Sistem Informasi. Ilmu Lahir pada 05 Januari 1994 di Desa Ujung Seribu 1 Kecamatan Tinggi Raja, Kabupaten Asahan, Sumatera Utara. Lulus S-1 Program Studi Sistem Informasi di Sekolah Tinggi Manajemen Informatika dan Komputer Royal Kisaran tahun 2015. Kemudian melanjutkan program Pascasarjana S-2 di Universitas Putra Indonesia YPTK Padang dan lulus pada tahun 2019 Program Studi Teknik Informatika Konsentrasi Sistem Informasi. Pernah bekerja sebagai Team Leader di Nestle Indonesia cabang padang sebelum akhirnya memantapkan diri menjadi Dosen hingga saat ini dan diperbantukan menjadi Sekretaris Prodi pada Program Studi Pendidikan Guru Pendidikan Anak Usia Dini (PG-PAUD) Universitas Adzkia.

Muhammad Thoriq, S.Kom., M.Kom. Lahir di Kota Padang Sumatera Barat pada tanggal 05 Oktober 1992. Dalam menempuh Pendidikan dimulai dari Sekolah Dasar SD Kartika 1-11 Padang tamat tahun 2004, SMPN 9 Padang Selama 1 tahun lalu dilanjutkan ke SMPN 8 padang tamat pada tahun 2007 dan di SMAS Semen Padang tamat pada tahun 2010. Lalu melanjutkan ke pendidikan tinggi yaitu Sarjana (S1) Universitas Putra Indonesia YPTK Padang, lulus pada tahun 2014 dengan jurusan Sistem Komputer. Kemudian melanjutkan program Pascasarjana (S2) di Universitas Putra Indonesia YPTK Padang dan lulus pada tahun 2019 Program Studi Teknik Informatika Konsentrasi Teknologi Informasi. Penulis mengabdikan diri sebagai salah satu Dosen di Bidang Ilmu Komputer khususnya Program Studi Informatika di Universitas Adzkia dan menjadi Dosen tetap pada maret tahun 2021 di kampus tersebut. Saat ini menjadi bagian Struktural di Universitas Adzkia sebagai Ketua Program Studi Informatika.

Aldo Eko Syaputra, S.Kom., M.Kom. Lahir di Kota Depok Provinsi Jawa Barat pada tanggal 07 Juli 1996. Dalam menempuh Pendidikan dimulai dari Sekolah Dasar SDN 03 Taruang-Taruang Tamat 2008, MTsN Sungai Lasi tamat tahun 2011, SMAN 1 Sungai Lasi tamat tahun 2014. Lalu melanjutkan ke pendidikan tinggi swasta yaitu Sarjana (S1) Universitas Putra Indonesia YPTK Padang, lulus pada tahun 2018 dengan jurusan Sistem Informasi. Kemudian melanjutkan program Pascasarjana (S2) di Universitas Putra Indonesia YPTK Padang dan lulus pada tahun 2019 Program Studi Teknik Informatika Konsentrasi Sistem Informasi. Penulis mengabdikan diri

sebagai salah satu Dosen di Bidang Ilmu Komputer khususnya Program Studi Sistem Informasi di Universitas Adzkia dan menjadi Dosen tetap pada maret tahun 2021 di kampus tersebut.

Dasar-Dasar Sistem Basis Data



Buku ini "Dasar-dasar Sistem Basis Data" merupakan panduan yang komprehensif untuk memahami konsep, teori, dan praktik terkait dengan sistem basis data dalam konteks dunia teknologi informasi yang terus berkembang pesat.

PT. JASA NIAGA DIGITAL INDONESIA
Jl. H. Maulana Link. VI, Labuhan Ruku
Talawi, Batu Bara, Sumatera Utara
21254

ISBN 978-623-10-0276-1 (PDF)
9 786231 002761